(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 163 434 A1**

(12) **EUROPEAN PATENT APPLICATION**
published in accordance with Art. 153(4) EPC

(43) Date of publication:
03.05.2017 Bulletin 2017/18

(21) Application number: **15810966.0**

(22) Date of filing: **25.06.2015**

(51) Int Cl.:
G06F 9/44 *(2006.01)*

(86) International application number:
PCT/CN2015/000454

(87) International publication number:
WO 2015/196783 (30.12.2015 Gazette 2015/52)

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR
Designated Extension States:
BA ME
Designated Validation States:
MA

(30) Priority: **25.06.2014 CN 201410291096**

(71) Applicant: **Chengdu Puzhong Software Limited Company**
Chengdu, Sichuan 610041 (CN)

(72) Inventors:
- **FU, Changming**
  Chengdu
  Sichuan 610057 (CN)
- **LONG, Chunsheng**
  Chengdu
  Sichuan 610041 (CN)
- **TANG, Hong**
  Chengdu
  Sichuan 610045 (CN)

(74) Representative: **Gallego Jiménez, José Fernando**
Ingenias Creaciones, Signos e Invenciones S.L.P.
Avda. Diagonal 421, 2
08008 Barcelona (ES)

(54) **SOFTWARE ELEMENT MODEL-BASED UNIVERSAL SOFTWARE MODELLING METHOD FOR CONSTRUCTING SOFTWARE MODEL**

(57) Software element model based univeral software modeling method to construct software model. After determining basic constituents of the software element model, the present invention constructs software models through the software hierarchy model, the software interface models, the software interaction models, the software algorithm model, the software process model, and the software transfer model as step features and thereby provides a specification for software modeling in various fields; such specification has advantages including generality and convenience; the software models constructed through the present invention is executable, has a clear structure, adjustable hierarchies, and controllable granularities for modeling; this method supports both top-down analysis and bottom-up integration for modeling in various software systems. The quantity of required elements for modeling is small and the modeling method is simple, thereby even those not skilled in any modeling language nor computer programming language can easily and independently construct the software model, eliminating the tedious and unnecessary communication with and dependence on professional modelers and application developers, greatly reducing modeling time.

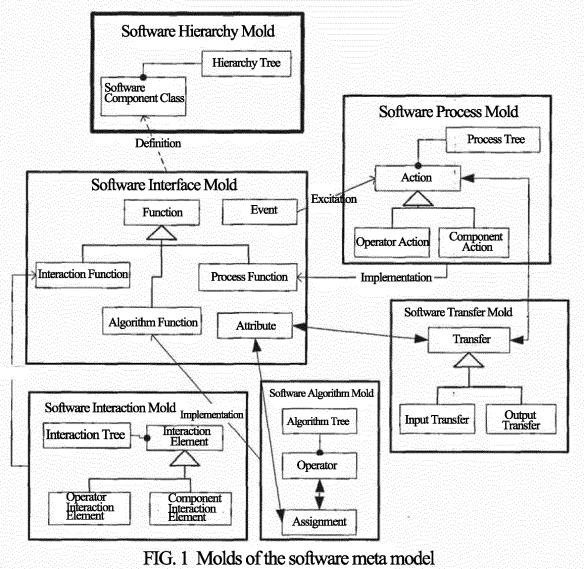

FIG. 1 Molds of the software meta model

EP 3 163 434 A1

Description

TECHNICAL FIELD

[0001] The present invention relates to the technical field of software engineering, and more specifically, a software element model based univeral software modeling method to construct software model and computer program products thereof.

TECHNICAL BACKGROUND

[0002] Nowadays, with rapidly developing information technology, softwares are playing an increasingly important role in our living and development. The object-oriented software method has become quite mature and mainstream after the function-oriented method, process-oriented method etc. However, many inadequacies for the object-oriented method has exposed as software systems become more complex and thus a model-driven method has become a pursued ideal software designing method. A core idea of the model-driven software method is in that software developers focus on constructing software models and representing knowledges as the software models and computers automatically convert the software models into executable program codes. Although the idea of the model-driven method has been proposed long before, in the industry, it is substantially present in a level of concept, slogan, and exploring even after last two decades of efforts, so that the object-oriented method still is a mainstream in actual applications. The most basic reasons for slow developing of the model-driven method is considered being lack of a method capable of supporting to simply and effectively construct software models. In fact, the software modeling is the core activity for software developing despite software methods.

[0003] In practical activities for the software modeling, users found that these software models share many common constructions where these constructions can be described by a model called the software element model. The software element model is a model describing software models; it is an abstraction of the software models and provides a set of basic elements and rules required for constructing software models, which is used as a specification for software modeling and thus plays a decisive role in the quality of the software models and the efficiency of software modeling.

[0004] Thus, study of the software element model has been a major focus. The MOF, which is proposed by OMG and is widely recognized in the industry, is systemic study on the software modeling. The MOF includes a set of four layers of model descripting mechanisms of which layer M1 is the software element model. The UML (Unified Modeling Language), which belongs to the M1 layer, is the current mainstream industry standard for object oriented visual modeling language. In fact, even though the UML clearly states that it neither is a methodology nor provide any software modeling method and is only a graphical descriptive language for descripting modeling methods to cooperate the software modeling methods, it is used as the software element model to a large extent. Unfortunately, the UML has the following drawbacks: Firstly, it lacks of the executability: although the UML states that it provides a great deal of flexibility for constructing models, the UML is substantially lacks a rigorous theoretical support for modeling, which results in that it cannot be assured that integral and consistency models are obtained by using the UML. The software model constructed by using UML is lack of executability, i.e., lack of sufficiency and consistency information by which the software model constructed by using UML is automatically converted into a software executable by computers, and thus the yielded software must be manually edited through codes in order to be executable by computers. This drawback further makes the UML only be a supplementary expression tool for software models rather than a true software element model. Secondly, it is code- oriented: although the UML states it is a model descripting language independent of specific languages and platforms, it is substantially oriented towards to expression tools by which the programmers employ object-oriented programming languages in analyzing and designing programs. Such a programmer-oriented code directing makes "paying great attention to the realization, and paying little attention to the service" have become an industry ill in the software developing. Thirdly, it is difficult to use: the UML creates a lot of concepts, relations and diagrams. Relationships among these concepts, relations, and diagrams are loose and numerous. The UML is originally designed for programmers. However, the UML's complication and disorder are not only hard for programmers to grasp, but also even more difficult for industry experts to understand, far from satisfying the needs of software modeling.

[0005] With a gradual rise of knowledge engineering, ontology element model recently is becoming a hot topic of research. Ontology is defined as an explicit specification of a conceptual model. The ontology element model effectively determines concepts commonly recognized in an industry with concepts as core elements, formal language as means of description, and formal logic as reasoning mechanisms, and gives clear definitions of the relations among these concepts. The ontology element model focuses on establishment and application of concept system and on classification, expression, and reasoning of information. The modeling primitive for the theory of the ontology element models includes concepts, relations, functions, axioms, and instances, providing a theoretical framework from the view of knowledge management: the international standard ISO /IEC19763 (interoperable element model framework MFI) provides a management specification with concept ontology and process ontology as core knowledge and information sharing specification with an ontology registration element model and a process registration element model

as cores; Chinese patent filing 200610125050.8 provides an application method for WEB service oriented industry requirement modeling based on ISO/IEC 19763's ontology registration element model, process registration element model and ontology description tool Protege. From the software modeling perspective, first, the ontology element model focuses on knowledge management and information sharing rather than universal software modeling; second, ontology element model employs inexplicable formal language, which is difficult for personnel.

[0006] In all, though the model-driven method has been generally recognized as an ideal software developing method, the software element model that is easy for ordinary industry personnel to grasp, provides a universal software modeling norm, supports software modeling in all fields, and constructs executable models so that the software models are automatically converted into the executable software of the computer by an automatic code generating technology to implement the model-driven software method, currently is still lacking and in demand.

SUMMARY OF THE INVENTION

[0007] In view of the above drawbacks of the prior art, the objective of the present invention is to provide a software element model based univeral software modeling method to construct software model, by which the constructing of the software model is accomplished with a software hierarchy model, software interface models, software interaction models, software algorithm models, software process models, and software transfer models as step elements, after determining basic constituents of the software element model; the software models in various specific fields are constructed by using basic molds in interdisciplinary fields; and a universal software modeling specification easily understood and grasped by those skilled in this art is provided for modeling activities in various fields.

[0008] The objective of the present invention is achieved by the following means.

[0009] A software element model based univeral software modeling method to construct software model, by means of a computer readable storage medium having a computer readable program code stored therein, the computer readable program code containing instructions executable by a processor of a computer software to implement a method of constructing software model by processing data conforming to the software element model and describing the software model, the software model describing a software system, the software element model comprising:

a software hierarchy mold which describes the software hierarchy model of the software model in a tree structure whose nodes are software component classes and which is used as a template to be configured in an actual software modeling environment to form the software hierarchy model of the software model, wherein the software hierarchy model refers to the hierarchy relationships constituted by the software component classes as the nodes in the software model, wherein the software component class refers to a set of software component instances with the same external features, and wherein the tree structure, whose nodes are the software component classes, is referred as a hierarchy tree;

a software interface mold which describes software interface models by an optional structure of an attribute set, a function set, and an event set, the software interface mold is used as a template in the actual software modeling environment to be configured to form the software interface models, wherein the software interface models refer to external features of the software component classes, wherein the functions in the function set include software interaction functions, software algorithm functions, and software process functions, wherein the software interaction function is implemented by a software interaction model, wherein the software algorithm function is implemented by a software algorithm model, and wherein the software process function is implemented by a combination of software process models and software transfer models;

a software interaction module which describes the software interaction models by a tree structure whose nodes are interaction elements and which is used as a template in the actual software modeling environment to be configured to form the software interaction models, wherein the software interaction model refers to a description of a way for implementing the software interaction function with a combination of the interaction elements and the interaction element refers to a functional element for interacting information with the actual software modeling environment;

a software algorithm mold which describes software algorithm models by a tree structure whose nodes are operators and which is used as a template in the actual software modeling environment to be configured to form the software algorithm model, wherein the software algorithm model refers to a description of the algorithm which implements the software algorithm function by using the combination of operators, and wherein the operator refers to a component with a previously realized specific function;

a software process mold which describes software process models by combining actions as nodes and which is used as a template in the actual software modeling environment to configure the software process models, wherein the software process model refers to a description of a way of the software process function which is implemented using a combination of the actions and wherein the action refers to an execution of a function;

a software transfer mold which describes software transfer models by a transfer set and which is used as a template in the actual software modeling environment to be configured to form the software transfer models, wherein the software transfer model refers to transfer relationships of the data of involved actions and a transfer in the transfer set is a transfer relationship of the data between one attribute and another attribute;

specific steps to construct the software model described by the six molds are as follows:

1) constructing the software hierarchy model: the software hierarchy mold reading in software hierarchy model commands from the actual software modeling environment, wherein the software hierarchy model command refers to command such as creating a software component class, adding a software component class, selecting a software component class, naming a software component class, or deleting a software component class for the hierarchy tree and wherein the software hierarchy mold performs corresponding operations on the software component class nodes in response to the software hierarchy model commands to obtain the software hierarchy model;

2) constructing the software interface models: constructing the software interface model for each software component class of the software hierarchy model obtained in the step 1), the steps for constructing each software interface model including: the software interface mold reading in software interface model commands from the actual software modeling environment, wherein the software interface model command refers to command such as creating, naming, or deleting the attributes, the functions, and the events, wherein the software interface mold performs corresponding operations in response to the software interface model commands to obtain the software interface model, and wherein the software interaction models for implementing software interaction functions are constructed by step 3), wherein the software algorithm models for implementing software algorithm functions are constructed by step 4), and wherein the software process models for implementing software process functions are constructed by the step 5);

3) constructing the software interactive models: constructing the software interaction model for each software interaction function obtained in the step 2), steps for constructing each software interaction model including: the software interaction mold reading in software interaction model commands from the actual software modeling environment;

4) constructing the software algorithm models: constructing the software algorithm model for each software algorithm function obtained in the step 2), the steps for constructing each software algorithm model including: the softweare algorithm mold reading in software algorithm model commands from the actual software modeling environment;

5) constructing the software process models: constructing the software process model for each software process function obtained in the step 2), the steps for constructing each software process model including: the software process mold reading in software process model commands from the actual software modeling environment; and

6) constructing the software transfer models: constructing the software transfer model for each action in the software process models obtained in the step 5), the steps for constructing each software transfer model including: the software transfer mold reading in software transfer model commands from the actual software modeling environment, wherein the software transfer model command refers to the command such as adding a transfer, adding a transfer, or deleting a transfer and wherein the software transfer mold performs corresponding operations in response to the software transfer model commands to obtain the software transfer model.

[0010] Thereby, the software model constructed by the software hierarchy model, the software interface models, the software algorithm models, the software process models, and the software transfer models is accomplished.

[0011] The software element model applies the following modeling rules:

a combination of the software process mold and the software transfer mold provide a universal means to describe and configure functions; the software interaction mold provides a simplified alternative for replacing the combination of the software process mold and the software transfer mold if only interaction elements are used to implement the functions;

a combination of the software process mold and the software transfer mold provide a universal means to describe and configure functions; the software algorithm mold provides a simplified alternative for replacing the combination of the software process mold and the software transfer mold if only operators are used to implement the functions;

the software element model employs a parent-child structure as a base recursive unit to recursively describe the software model; the parent-child structure refers to a structure of parent-child relationships in a hierarchy tree, constituted by an involved software

component class and all child software component classes thereof;

the specific function of the step 2) can only be any one of the software interaction function, the software algorithm function, and software process function;

the software interaction model commands for constructing the software interaction model in the step 3) refer to commands, such as adding an interaction element, selecting an interaction element, naming an interaction element, and deleting an interaction element, and the software interaction mold performs corresponding operations in response to the software interaction model commands to obtain the software interaction model; the interaction elements include operator interaction elements and component interaction elements; the operator interaction element refers to a component with a previously realized specific function and the component interaction element refers to one execution of the interaction function in a set of the interaction functions in the parent-child structure, the set of the interaction functions in the parent-child structure refers to a collection constituted by all interaction functions of the involved component class and all interaction functions of all child component classes thereof in the parent-child structure, the tree structure of which the nodes are the interaction elements is referred to as an interaction tree, the software interaction molds include specific forms of, for example, interface molds, file molds, database molds, and communicating molds; and the software interaction models include specific forms of interface models, file models, database models, and communicating models;

the software algorithm model commands for constructing the software algorithm model in the step 4) refer to commands, such as adding an operator, selecting an operator, naming an operator, and deleting an operator, as well as adding an assignment, selecting an assignment, and deleting an assignment, and the software algorithm mold performs corresponding operations in response to the software algorithm model commands to obtain the software algorithm model; the operators include logic operators with logic functions and computation operators with calculation functions; the tree structure whose nodes are operators is referred to as an algorithm tree; the assignment refers to an assignment relationship between two attributes in a set of the algorithm attributes; and the set of the algorithm attributes refers to a collection constituted by a set of attributes of the involved software component classes, a set of attributes of all operators, and a set of attributes of all interaction elements in the software interaction model;

the software process model commands for constructing the software process model in step 5) refer to commands such as adding an action, selecting an action, naming an action, and deleting an action and the software process mold performs a corresponding operation in response to the software process model commands to obtain the software process model; the actions include both component actions and operator actions; the component action refers to one execution of the functions in the function set in the parent-child structure, the function set in the parent-child structure refers to a collection constituted by the function set of the involved software component class and function sets of all child software component classes in the parent-child structure; the operator action refers to one execution of operator's function; the software process models include attribute process models and event process models, the software process mold includes attribute process mold and event process mold, and the attribute process mold describes an attribute process model through a process tree as the structure, which is a tree structure constituted by actions as nodes; the event process mold describes an event process model by a set of event associations as the structure; the event association in the set of event associations is an association relationship between an event in a set of events in a parent-child structure and an operator action or a component action; the event set in the parent-child structure is a collection constituted by the event set of the involved software component class and the event sets of all interaction elements in the interaction model thereof and the event sets of all child software component classes and the event sets of all interaction elements in the interaction model thereof, in the parent-child structure.

[0012] Besides action attributes which refers to the attribute of the component class where the action is, the attributes relevant to transfers are limited to the parent-child attribute set, which refers to a collection constituted by the attribute set of the involved software component classes and attribute sets of all child software component classes thereof in the parent-child structure.

[0013] Alternatively, the attribute sets in the parent-child structure may include the attribute sets of all interaction elements in the interaction models of the involved software component class and the attribute sets of all interaction elements in the interaction models of all child software component classes in the parent-child structure.

[0014] Thus, after determining basic constituents of the software element model, the present invention constructs software models through the software hierarchy model, the software interface models, software interaction models, the software algorithm models, the software process models, and the software transfer models as step features and thereby provides a specification for software modeling in various fields; such specification has advantages including generality and convenience; the software models constructed through the present invention are executable, have clear structures, adjustable

hierarchies, and controllable granularities; this modeling method supports both top-down analysis and bottom-up integration for modeling in various software system modeling; the quantity of required elements for modeling is small and the modeling method is simple, thereby even those not skilled in any modeling language nor computer programming language can easily and independently construct the software model, eliminating the tedious and unnecessary communication with and dependence on professional modelers and application developers, greatly reducing modeling time.

[0015] In summary, the present invention has obvious advantages over the prior art as follows:

(1) executability: the software model constructed according to the present invention is executable, that is, has an integrity and a full consistency in which the software model can be mapped to a program executable by a computer;

(2) generality: the software model constructed according to the present invention has a clear structure, adjustable hierarchies, controllable granularities, and generality suitable for all types of systems. That is, not only suitable for software algorithm modeling but also rapid prototyping of the system and even more suitable for large, complex software modeling; not only convenient for top-down analysis but also bottom up integration; not only suitable for software integration based on prefabricated units and software expansion based on customized units, but also suitable for distributed software interconnection and communication; not only suitable for practical engineering software modeling, but also suitable for various information software modeling; not only suitable for equipment information software simulation modeling, but also suitable for MIS (management information software) modeling; not only suitable for desktop software modeling, embedded device software modeling, mobile terminal software modeling, but also suitable for LAN software modeling, WAN software modeling, and cloud computation environmental software modeling; not only suitable for applied software modeling, but also suitable for software development environment modeling; and

(3) ease of use: the elements of the present invention are concise, rules thereof are simple, and methods thereof are universal. Even those not skilled in any complex modeling language nor any computer programming language can easily take advantage of the present invention to construct the software model with executability in a relatively short period of time, eliminating the tedious and unnecessary communication with and dependence on professional modelers and application developers, enabling the resultant software model to be more fitted to the expectations of those skilled in this art, and eliminating possible understanding errors of the professional modelers or the application developers; at the same time, because communication time is saved, modeling time is greatly reduced.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016]

Fig. 1 is molds of the software element model.
Fig. 2 is construction steps of a software model.
Fig. 3 is a software interaction model.
Fig. 4 is assignment rules in a software algorithm model.
Fig. 5 is actions and function sets in a parent-child structure.
Fig. 6 is transfers and attribute sets in a parent-child structure.
Fig. 7 is a block diagram of a computer for implementing software element model based univeral software modeling method to construct software model.
Fig. 8 is the business management YWGL hierarchy model.
Fig. 9 is the business management YWGL interface model.
Fig. 10 is the sales management XSGL interface model.
Fig. 11 is the production management SCGL interface model.
Fig. 12 is the purchase management CGGL interface model.
Fig. 13 is the business assistant YWZS interface model.
Fig. 14 is the distributed sales product FXP interface model.
Fig. 15 is the direct sales product ZXP interface model.
Fig. 16 is the main parts ZJ interface model.
Fig. 17 is the auxiliary parts LJ interface model.
Fig. 18 is the finished product CP interface model.
Fig. 19 is the business display interaction model.
Fig. 20 is the sales display interaction model.
Fig. 21 is the production display interaction model.
Fig. 22 is the purchase display interaction model.
Fig. 23 is the distributed sales display interaction model.
Fig. 24 is the direct sales display interaction model.
Fig. 25 is the main parts display interaction model.
Fig. 26 is the auxiliary parts display interaction model.
Fig. 27 is the finished product display interaction model.
Fig. 28 is the product type configuration algorithm model.
Fig. 29 is the sales update algorithm model.
Fig. 30 is the production delivery algorithm model.
Fig. 31 is the production update algorithm model.
Fig. 32 is the purchase implementation algorithm model.

Fig. 33 is the purchase update algorithm model.
Fig. 34 is the distributed sales update algorithm model.
Fig. 35 is the direct sales update algorithm model.
Fig. 36 is the main parts update algorithm model.
Fig. 37 is the auxiliary parts update algorithm model.
Fig. 38 is the finished product update algorithm model.
Fig. 39 is the main parts processing algorithm model.
Fig. 40 is the main parts delivery algorithm model.
Fig. 41 is the auxiliary parts processing algorithm model.
Fig. 42 is the auxiliary parts delivery algorithm model.
Fig. 43 is the auxiliary parts receipt algorithm model.
Fig. 44 is the finished product assembly algorithm model.
Fig. 45 is the main business procedure process model.
Fig. 46 is the configuration display and control process model.
Fig. 47 is the business configuration process model.
Fig. 48 is the business instance creation process model.
Fig. 49 is the business instance configuration process model.
Fig. 50 is the business operation process model.
Fig. 51 is the operation display and control process model.
Fig. 52 is the sales display and control process model.
Fig. 53 is the sales operation process model.
Fig. 54 is the production display and control process model.
Fig. 55 is the production operation process model.
Fig. 56 is the production planning process model.
Fig. 57 is the production implementation process model.
Fig. 58 is the purchase display and control process model.
Fig. 59 is the purchase operation process model.
Fig. 60 is the distributed sales display and control process model.
Fig. 61 is the direct sales display and control process model.
Fig. 62 is the main procedure frame loop transfer model.
Fig. 63 is the main procedure condition transfer model.
Fig. 64 is the business operation state negation transfer model.
Fig. 65 is the sales instance creation transfer model.
Fig. 66 is the production instance creation transfer model.
Fig. 67 is the purchase instance creation transfer model.
Fig. 68 is the production configuration traversal transfer model.
Fig. 69 is the production serial number increment transfer model.
Fig. 70 is the production serial number assignment transfer model.
Fig. 71 is the purchase configuration traversal transfer model.
Fig. 72 is the purchase serial number assignment transfer model.
Fig. 73 is the sales serial number reset transfer model.
Fig. 74 is the sales configuration traversal transfer model.
Fig. 75 is the sales serial number assignment transfer model.
Fig. 76 is the sales-production configuration comparison transfer model.
Fig. 77 is the sales-production configuration condition transfer model.
Fig. 78 is the sales-production product name assignment transfer model.
Fig. 79 is the sales-purchase configuration comparison transfer model.
Fig. 80 is the sales-purchase configuration condition transfer model.
Fig. 81 is the sales-purchase product name assignment transfer model.
Fig. 82 is the sales-production operation transfer model.
Fig. 83 is the sales-purchase operation transfer model.
Fig. 84 is the sales receipt transfer model.
Fig. 85 is the shipment quantity summary transfer model.
Fig. 86 is the total shipment quantity summary transfer model.
Fig. 87 is the inventory quantity summary transfer model.
Fig. 88 is the contract quantity summary transfer model.
Fig. 89 is the demand quantity summary transfer model.
Fig. 90 is the order quantity summary transfer model.
Fig. 91 is the main-parts pending processing quantity summary transfer model.
Fig. 92 is the auxiliary parts pending processing quantity summary transfer model.
Fig. 93 is the parts receipt transfer model.
Fig. 94 is the finished product assembly transfer model.

[0017] In the drawings listed above, the figure showing transfer model emphasized the involved action's transfer model using boldface font. Meanwhile, for the convenience of describing reading, the non-boldface figures in the drawings represent information regarding involved actions.

DETAILED DESCRIPTION OF THE INVENTION

[0018] Generally, a computer comprises a central processor, a memory, an Input and Output (I/O) interface, and a bus; and furthermore, the computer is connected with an input and output device and a storage medium. The central processor takes charge of functions of computing and controlling the computer. The central processor may only include one central processing unit, or may include a plurality of central processing units distributed at one or more positions.

[0019] The memory medium may be formed by any known computer readable storage medium. For example, a buffer memory may temporarily store some program codes so as to reduce time for extracting codes from a large-capacity memory when the program is run. In the meantime, the memory may reside at a certain physical position, and may be stored in one or more types of data, or may be distributed in different physical systems in different forms. Moreover, the memory may also be distributed in a Local Area Network (LAN) or a Wide Area Network (WAN). The memory may contain program codes for implementing a general modeling method to establish a system view based on a system element view, or may contain other codes not shown in the diagram, such as an operating system.

[0020] The input and output interface allows the computer to exchange information with the storage medium or another computer. The input and output device contains any known external device type, such as a display device, a keyboard, a mouse, a printer, a sound box, a handheld device, and a facial mask, etc. The bus provides communication connection among respective component parts inside the computer, including a variety of transmission connection forms such as electrical, optical, wireless and other forms. The storage medium includes any known computer readable storage medium, such as a magnetic disc, an optical disc, etc. The storage medium may contain one or more examples of a general system view established by the system element view.

[0021] A person skilled in the art can know that the present invention can be implemented as an all hardware product, an all software product, or a combination of hardware and software, which is commonly referred to as a module. Moreover, the present invention can be implemented by a computer program product stored in the computer readable medium. The computer readable medium may be, for example, but not limited to an electrical, a magnetic, an optical, an electromagnetic, an infrared or a semiconductor system, apparatus, or device or any combination of the above, and more particularly, the computer readable medium includes, but not limited to, the following: a random access memory (RAM), a read-only memory (ROM), an erasable and programmable read-only memory (EPROM or flash memory), a CD-ROM, an optical storage device, a magnetic storage device, and any combination of the above.

[0022] The computer program codes for implementing the method of the present invention can be programmed by one or more programming languages, including, for example, Java, Small, C++, C# and so on, and other process-oriented programming languages such as C. The program codes can be run on a personal computer, a handheld device, or an LAN or WAN.

[0023] A person skilled in the art surely knows that the method of the present invention may also be expressed by graphical representations, and such graphical representations all can be implemented as computer program codes, which can be processed by a general-purpose computer, a special-purpose computer and other programmable data processing apparatuses, to achieve the functions indicated by these graphical representations.

[0024] In the following embodiments, in order to maintain the completeness of description of a software model, the transfer models for all actions are listed. Among these, some of the actions do not require transfers of the data, and thereby the content of such a transfer model is null which will be expressed as the word "null".

[0025] Below, a further detailed description of the present invention will be given in conjunction with a specific embodiment in which constructing a business management YWGL software model by using the present invention is demonstrated. It should be known by those skilled in this art that the technical scope of the present invention is not limited to the following demonstrated contents of this embodiment.

Embodiment: constructing the business management YWGL software model

[0026] The present embodiment supposes that the firm's business mode is to profit from selling its own products and outsourcing products and models the business management software for achieving the following business management intentions:

(1) clearly distinguishing three modules: production management, purchase management, and sales management;

(2) configuration: configuring produced product type quantities and purchased product type quantities by an interface and configuring sales product type quantities from the produced product type quantities and the purchased product type quantities; and

(3) execution function: the sales management module real-time interacts with an actual application environment for contract order quantities and shipment quantities of the direct sales and distributed sales of each type of products, receives delivery information from the production management module and the purchase management module, and issues order information to the production management module and the purchase management module based on the sales status; the production management module and the purchase management module receive order information from the sales management mod-

ule, receive response order information after completed information from the actual application environment, start an internal process, and submit the delivery information to the sales management module. Each module real-time displays information by an interface.

[0027] The detailed procedure for constructing this embodiment's software model is given below.

Constructing the software hierarchy model

[0028] Firstly, for the convenience of describing understanding, Fig. 8 shows a completed software hierarchy model of the business management YWGL software model.

[0029] At the initial state before start of modeling, the software hierarchy mold creates a root node for a hierarchy tree, wherein the software component class of the root node is referred to as a root software component class;

the software hierarchy mold receives the command to select root software component class from the actual software modeling environment and sets the root software component class to the involved software component class in response to the foregoing command; the software hierarchy mold receives the command to modify software component class' name to "business management YWGL" from the actual software modeling environment and modifies the name of the root software component class to "business management YWGL" in response to the foregoing command; the involved software component class is referred to as a business management YWGL software component class in accordance with the name of the root software component class and name for the other software component classes may be deduced by analogy, which will not be repeated below;

the software hierarchy mold receives the command to set the component instance quantity to 1 from the actual software modeling environment and sets the component instance quantity of the business management YWGL software component class to 1 in response to the foregoing command;

the software hierarchy mold receives the command to add a child software component class from the actual software modeling environment and adds a child software component class for business management YWGL software component class in response to the foregoing command; the software hierarchy mold sets the foregoing child software component class to the involved software component class; the software hierarchy mold receives commands, from the actual software modeling environment, to modify the name of the involved software component class to "sales management XSGL" and modifies the name of the involved software component class to "sales management XSGL" in response to the foregoing commands;

in the foregoing steps, the software hierarchy mold receives the command from the actual software modeling environment to select the business management YWGL software component class and adds, in response to the foregoing command, three child software component classes, i.e., a production management SCGL software component class whose software component instance quantity is 0, a purchase management CGGL software component class whose software component instance quantity is 0, and a business assistant YWZS software component class whose software component instance quantity is 1, to the business management YWGL software component class:

in the foregoing steps, the software hierarchy mold receives the command to select the sales management XSGL software component class from the actual software modeling environment and add, in response to the foregoing command, two child software component classes, i.e., distributed sales product FXP software component class and direct sales product ZXP software component class, whose software component instance quantities are 1, to the sales management XSGL software component class; and

in the foregoing steps, the software hierarchy mold receives the command to select the production management SCGL software component class from the actual software modeling environment and adds, in response to the foregoing command, three child software component classes, i.e., main parts ZJ software component class whose instance quantity is 1, auxiliary parts LJ software component class whose instance quantity is 1, and finished product CP software component class whose instance quantity is 1, to the production management SCGL software component class.

[0030] So far, the software hierarchy model of the business management YWGL software model is accomplished.

Constructing the software interface models

[0031] Next, the procedure for constructing the software interface model for each of the software component classes in the software hierarchy model as described above will be given. The software interaction in this embodiment is demonstrated with interface displays as examples and thus it should be appreciated by those skilled in this art that the software interactions in other forms can be similarly handled by taking advantage of the present invention.

Business management YWGL software interface model

[0032] Fig. 9 shows a completed software interface model for business management YWGL software com-

ponent class, which is shortly referred to as the business management YWGL software interface model, names of other software interface models of the software component classes may be deduced by analogy; the procedure for constructing business management YWGL software interface model is as follows:

the software hierarchy mold receives the command to select the business management YWGL software component class from the actual software modeling environment and sets the business management YWGL software component class to the involved software component class in response to the foregoing command;
the software interface mold receives the command to add an attribute from the actual software modeling environment and adds a new attribute to the business management YWGL software component class in response to the foregoing command; the software interface mold sets the foregoing new attribute to the involved attribute; the software interface mold receives the command to modify the data type of the involved attribute as "bool" from the actual software modeling environment and modifies the data type of the involved attribute to "bool"; the software interface mold receives the command to modify the attribute name of the involved attribute to the word "main loop state" from the actual software modeling environment and modifies the attribute name of the involved attribute to the main loop state in response to the foregoing command; the attribute whose attribute name is the main loop state is shortly referred to as the main loop state attribute, and names for the subsequent attributes may be deduced by analogy, which will not be repeated below; and the software interface mold receives the command to set the attribute value of the involved attribute to "true" from the actual software modeling environment and sets the attribute value of the main loop state attribute to "true" in response to the foregoing command;
in the foregoing steps, the software interface mold adds the following attributes to the business management YWGL interface model: a business operation state attribute whose data type is the "bool" type and the attribute value is the "true"; a production product type quantity attribute whose data type is the integer and the attribute value is 3; a purchase product type quantity attribute whose data type is the integer and the attribute value is 2; a sales product type quantity attribute whose data type is the integer and the attribute value is 0;
the software interface mold receives the command to add a function from the actual software modeling environment and adds a software interaction function to the business management YWGL software component class in response to the foregoing command; the software interface mold sets the newly added software interaction function to the involved function; the software interface mold receives the command to modify the function name of the involved function to the word "business display" from the actual software modeling environment and modifies the function name of the involved function to the business display in response to the foregoing command, wherein such a function whose function name is referred to as a business display is shortly referred to as the business display interaction function, short names for the subsequent functions may be deduced by analogy, which will not be repeated below; and
in the foregoing steps, the software interface mold adds the following software process functions to the business management YWGL interface model: an operation display and control process function, a main business procedure (which is the entry function of the software model) process, a business configuration process function, a business operation process function, an instance creation process function, an instance configuration process function, a configuration display and control process function, and a product type configuration process function.

[0033] So far, the business management YWGL interface model is accomplished.

Sales management XSGL interface model

[0034] Fig. 10 shows a completed sales management XSGL interface model whose construction process is similar to that of the "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a product name attribute whose data type is "string" and attribute value is "sales product"; a product serial number attribute whose data type is "int" and attribute value is "1"; an inventory quantity attribute whose data type is "int" and attribute value is "0"; a minimum inventory quantity attribute whose data type is "int" and attribute value is "0"; a contract order quantity attribute whose data type is "int" and attribute value is "0"; a receipt quantity attribute whose data type is "int" and attribute value is "0"; an order quantity attribute whose data type is "int" and attribute value is "0"; a shipment quantity attribute whose data type is "int" and attribute value is "0"; a total shipment quantity attribute whose data type is "int" and attribute value is "0"; and a demand quantity attribute whose data type is "int" and attribute value is "0"; and
the function set contains three process functions: an internal order process function, a sales shipment process function, and a sales order process function.

Production management SCGL interface model

[0035] Fig. 11 shows a completed production management SCGL interface model whose construction process

is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a product name attribute whose data type is "string" and attribute value is "self-developed product"; a product serial number attribute whose data type is "int" and attribute value is "1"; an order quantity attribute whose data type is "int" and attribute value is "0"; a processed quantity attribute whose data type is "int" and attribute value is "0"; a delivery quantity attribute whose data type is "int" and attribute value is "0"; and a total delivery quantity attribute whose data type is "int"and attribute value is "0"; and
the function set contains three process functions: a production planning process function, a production implementation process function, and a production delivery process function.

Purchase management CGGL interface model

[0036] Fig. 12 shows a completed purchase management CGGL interface model whose construction process is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a product name attribute whose data type is "string" and attribute value is "purchased product"; a product serial number attribute whose data type is "int" and attribute value is "1"; a pending purchase quantity attribute whose data type is "int" and attribute value is "0"; a purchased quantity attribute whose data type is "int" and attribute value is "0"; a delivery quantity attribute whose data type is "int" and attribute value is 0; and a total delivery quantity attribute whose data type is "int" and attribute value is "0" the; and
the function set contains two algorithm functions, i.e., a purchase implementation algorithm function and a purchase delivery algorithm function.

Business assistant YWZS interface model

[0037] Fig. 13 is the accomplished business assistant YWZS interface model whose construction procedure is similar to that of the "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a product serial number attribute whose data type is the integer and the attribute value is 0; a constant zero attribute whose data type is the integer and the attribute value is 0; and a comparison result attribute whose data type is the "bool" type and the attribute value is "true".

Distributed sales product FXP interface model

[0038] Fig. 14 shows a completed distributed sales product FXP interface model whose construction procedure is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a minimum inventory quantity attribute whose data type is "int" and attribute value is "5"; a contract order quantity attribute whose data type is "int" and attribute value is "12"; and a shipment quantity attribute whose data type is "int" and attribute value is "8".

Direct sales product ZXP interface model

[0039] Fig. 15 shows a completed direct sales product ZXP interface model whose construction procedure is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a minimum inventory quantity attribute whose data type is "int" and attribute value is "6"; a contract order quantity attribute whose data type is "int" and attribute value is "3"; and a shipment quantity attribute whose data type is "int" and attribute value is "4".

Main parts ZJ interface model

[0040] Fig. 16 shows a completed main parts ZJ interface model whose construction procedure is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a main parts name attribute whose data type is "string" and attribute value is "main parts"; a pending processing quantity attribute whose data type is "int"and attribute value is "0"; a processed quantity attribute whose data type is "int"and attribute value is "0"; a delivery quantity attribute whose data type is "int"and attribute value is "0"; and a total delivery quantity attribute whose data type is "int" and attribute value is "0"; and
the function set contains two algorithm functions: a main parts processing algorithm function and a main parts delivery algorithm function.

auxiliary parts LJ interface model

[0041] Fig. 17 shows a completed auxiliary parts LJ interface model whose construction procedure is similar to that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: an auxiliary parts name attribute whose data type is "string" and attribute value is "auxiliary parts"; a pending processing quantity attribute whose data type is "int"and attribute value is "0"; a processed quantity whose data type is "int" and attribute value is "0"; a delivery quantity attribute

whose data type is "int" and attribute value is "0"; and a total delivery quantity attribute whose data type is "int" and attribute value is "0"; and

the function set contains a software interaction function, i.e., an auxiliary parts display algorithm function, and three algorithm functions: an auxiliary parts processing function, an auxiliary parts delivery algorithm function, and an auxiliary parts update algorithm function.

Finished product CP interface model

[0042] Fig. 18 shows a completed finished product CP interface model whose construction procedure is similar that of "business management YWGL interface model" and the content thereof is as follows:

the attribute set contains: a finished product name attribute whose data type is "string" and attribute value is "finished product"; a pending processing quantity attribute whose data type is "int" and attribute value is "0", a processed quantity attribute whose data type is "int"and attribute value is "0"; a single set main parts quantity attribute whose data type is "int" and attribute value is "2"; a single set auxiliary parts quantity attribute whose data type is "int"and attribute value is "6"; a main parts inventory quantity attribute whose data type is "int" and attribute value is "0"; a main parts receipt quantity attribute whose data type is "int" and attribute value is "0"; a parts inventory quantity attribute whose data type is "int" and attribute value is "0"; and a parts receipt quantity attribute whose data type is "int" and attribute value is "0"; and

the function set contains a product display software interaction function and three algorithm functions: a parts receipt algorithm function, a finished product assembly algorithm function, and a finished product update algorithm function.

Constructing the software interaction models

[0043] Next, the construction procedure of each of the software interaction models will be described in detail.

Business display software interaction model

[0044] Fig. 19 shows the accomplished software interaction model for implementing the business display functions in the business management YWGL interface model, which is shortly referred to as the business display interaction model. Short names of the software interaction models in the other interface models may be deduced by analogy, which will not be repeated below. The construction procedure of the business display interaction model is as follows:

the software interface mold receives and responds to the command from the actual software modeling environment to set the business display interaction function of the business management YWGL software component class to the involved function;

the software interaction mold receives and responds to the command from the actual software modeling environment to successively accomplish the following operations: creating a window element shortly referred to as a business management window element and a free layout element as a root interaction element of the business management window element, which is referred to as a root layout element of the business management;

in the foregoing steps, adding a tag element as a child interaction element for the root layout element of the business management and setting a text content attribute of the involved tag element as the word "business management interface", wherein the tag element is referred to as a business management interface tag element for the convenience for the description and names for other elements may be deduced by analogy, which will not be repeated below;

the software interaction mold receives and responds to the command from the actual software modeling environment to add a stack layout element, shortly referred to as a business configuration stack layout element, for the business management root layout element; the software interaction mold receives and responds to the command from the actual software modeling environment to add the following operator interaction elements as child interaction elements thereof: adding a tag element whose background color is set to gray and the text is "production product type quantity "; adding a textbox element whose text content is set to 3, shortly referred to as a production product type quantity textbox element; adding a tag element whose background color is set to gray and the text is "purchase product type quantity "; add a textbox element whose text is set to 2, shortly referred to as a purchase product type quantity textbox element; adding a stack layout element whose width value is set to 50; adding a button element whose text content attribute is set to "business configuration", shortly referred to as a business configuration button element;

the software interaction mold receives and responds to the command from the actual software modeling environment to add a card element composed of three card pages for the business management root layout element;

in the foregoing steps, selecting from the card elements a first card page shortly referred to as a production management card page, whose page name is changed into "production management"; adding, as a root interaction element of the production management card page, a free layout element which is shortly referred to as the production management card page root interaction element; adding a com-

ponent interaction element based on the production display interaction function of the production management SCGL software component class as a child interaction element of the production management card page root interaction element;
in the foregoing steps, selecting from the card elements a second card page shortly referred to as a production management card page for the convenience for the description, whose page name is changed into "purchase management"; adding, as a root interaction element of the purchase management card page, a free layout element which is shortly referred to as the purchase management card page root interaction element for the convenience for the description; adding a component interaction element based on the purchase display function of the purchase management CGGL software component class as a child interaction element of the purchase management card page root interaction element;
in the foregoing steps, selecting from the card element a third card page shortly referred to as a sales management card page for the convenience for the description, whose page name is changed into "sales management"; adding, as a root interaction element of the sales management card page, a free layout element which is shortly referred to as the sales management card page root interaction element the convenience for the description; adding a component interaction element based on the sales display interaction function of the sales management XSGL software component class as a child interaction element of the sales management card page root interaction element.

[0045] So far, the business display interaction model is accomplished.

Sales display interaction model

[0046] As shown in Fig. 20, , the construction procedure of the sales display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a template of the sales display interaction model, a type template which is shortly referred to as the sales type template and contains an instance group layout element and an instance template, wherein the instance template is a display template, referred to as a sales instance template, for an instance of the sales management XSGL software component class and the instance group layout element as a container is a stack layout element, referred to as a sales instance group layout element, responsible for the display layout between all instances of the sales management XSGL software component class, where the stack direction of the sales instance group layout element is set to the longitudinal direction as a default;
adding, in the sales instance template, a stack layout element which is referred to as a sales instance stack layout element whose layout direction is set to the horizontal direction;
adding, in the sales instance stack layout element, a stack layout element which is referred to as a sales product name stack layout element; the software interaction mold receives and responds to the command from the actual software modeling environment to successively accomplish the following operations: setting the layout direction of the sales product name stack layout element to the horizontal direction; adding, in the sales product name stack layout element, a tag element whose text content is set to "sales products" and which is shortly referred to as the sales product name tag element; adding, in the sales product name stack layout element, a tag element whose text content is set to "1" and which is shortly referred to as the sales serial number tag element;
adding, in the sales instance stack layout element, a stack layout element which is referred to as a sales data stack layout element whose layout direction is set to the longitudinal direction as a default;
adding, in the sales data stack layout element, a stack layout element which is shortly referred to as the sales contract quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the sales contract quantity stack layout element, a tag element whose text content is "contract quantity" and the background color is set to gray; adding, in the sales contract quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the sales contract quantity display tag element;
adding, in the sales data stack layout element, a stack layout element which is referred to as a sales receipt quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the sales receipt quantity stack layout element, a tag element whose text content is "receipt quantity" and the background color is set to gray; adding, in the sales receipt quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the sales receipt quantity display tag element;
adding,, in the sales data stack layout element, a stack layout element which is referred to as a sales shipment quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the sales shipment quantity stack layout element, a tag element whose text content is "shipment quantity" and the background color is set to gray; adding, in the sales shipment quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the sales ship-

ment quantity display tag element;

adding, in the sales data stack layout element, a stack layout element which is referred to as a total sales shipment quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the total sales shipment quantity stack layout element, a tag element whose text content is "total shipment quantity" and the background color is set to gray; adding, in the total sales shipment quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the total sales shipment quantity display tag element; and

adding, in the sales instance stack layout element, a component interaction element based on the sales display function of the distributed sales product FXP software component class and adding, in the sales instance stack layout element, a component interaction element based on the sales display function of the direct sales product FXP software component class.

Production display interaction model

[0047] As shown in Fig. 21, , the construction procedure of the production display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a template of the production display interaction model, a type template which is shortly referred to as the production type template and contains an instance group layout element and an instance template, wherein the instance template is a display template, referred to as a production instance template, for an instance of the production management SCGL software component class and the instance group layout element as a container is a stack layout element, referred to as a production instance group layout element, responsible for the display layout between all instances of the production management SCGL software component class, where the stack direction of the production instance group layout element is set to the longitudinal direction as a default;

adding, in the production instance template, a stack layout element which is referred to as a production instance stack layout element whose layout direction is set to the horizontal direction;

adding, in the production instance stack layout element, a stack layout element which is referred to as a produced product name stack layout element; the software interaction mold receives and responds to the command from the actual software modeling environment to successively accomplish the following operations: setting the layout direction of the produced product name stack layout element to the horizontal direction; adding, in the produced product name stack layout element, a tag element whose text content is set to "produced products" and which is shortly referred to as the produced product name tag element; adding, in the produced product name stack layout element, a tag element whose text content is set to "1" and which is shortly referred to as the produced product serial number tag element;

adding, in the production instance stack layout element, a stack layout element which is referred to as a production data stack layout element whose layout direction is set to the longitudinal direction as a default;

adding, in the production data stack layout element, a stack layout element which is shortly referred to as the production order quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the production order quantity stack layout element, a tag element whose text content is "order quantity" and the background color is set to gray; adding, in the production order quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the production order quantity display tag element;

adding, in the production data stack layout element, a stack layout element which is referred to as a production completion quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the production completion quantity stack layout element, a tag element whose text content is "processed quantity" and the background color is set to gray; adding, in the production completion quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a production completion quantity display tag element;

adding, in the production data stack layout element, a stack layout element which is referred to as a production delivery quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the production delivery quantity stack layout element, a tag element whose text content is "delivery quantity" and the background color is set to gray; adding, in the production delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a production delivery quantity display tag element for the convenience for the description;

adding, in the production data stack layout element, a stack layout element which is referred to as a production total delivery quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the production total delivery quantity stack layout element, a tag element whose text content is "total delivery quantity" and the background color is set to gray; adding, in the production total delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a production total delivery quantity

display tag element for the convenience for the description;
adding, in the production instance layout element, a button element which is shortly referred to as the production completion button element whose content attribute is set to "production completion";
adding, in the production instance stack layout element, a component interaction element based on the main parts display function of the main parts ZJ software component class; adding, in the production instance stack layout element, a component interaction element based on the auxiliary parts display function of the auxiliary parts LJ software component class; and adding, in the production instance stack layout element, a component interaction element based on the finished product display function of the finished product CP software component class.

Purchase display interaction model

[0048] As shown in Fig. 22, , the construction procedure of the purchase display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a template of the purchase display interaction model, a type template which is shortly referred to as the purchase type template and contains an instance group layout element and an instance template, wherein the instance template is a display template, referred to as a purchase instance template, for an instance of the purchase management CGGL software component class and the instance group layout element as a container is a stack layout element, referred to as a purchase instance group layout element, responsible for the display layout between all instances of the purchase management CGGL software component class, where the stack direction of the purchase instance group layout element is set to the longitudinal direction as a default;
adding, in the purchase instance template, a stack layout element which is referred to as a purchase instance stack layout element whose layout direction is set to the horizontal direction;
adding, in the purchase instance stack layout element, a stack layout element which is referred to as a purchase product name stack layout element; the software interaction mold receives and responds to the command from the actual software modeling environment to successively accomplish the following operations: setting the layout direction of the purchase product name stack layout element to the horizontal direction; adding, in the purchase product name stack layout element, a tag element whose text content is set to "purchase products" and which is shortly referred to as the purchase product name tag element; adding, in the purchase product name stack layout element, a tag element whose text content is set to "1" and which is shortly referred to as the purchase product serial number tag element;
adding, in the purchase instance stack layout element, a stack layout element which is referred to as a purchase data stack layout element whose layout direction is set to the longitudinal direction as a default;
adding, in the purchase data stack layout element, a stack layout element which is shortly referred to as the pending purchasing quantity stack layout element; adding, in the pending purchasing quantity stack layout element, a tag element whose text content is "pending purchase quantity" and the background color is set to gray; adding, in the pending purchasing quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the pending purchasing quantity display tag element;
adding, in the purchase data stack layout element, a stack layout element which is referred to as a purchased quantity stack layout element; adding, in the purchased quantity layout element, a tag element whose text content is "purchased quantity" and the background color is set to gray; adding, in the purchased quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a purchased quantity display tag element;
adding, in the purchase data stack layout element, a stack layout element which is referred to as a purchase delivery quantity stack layout element; adding, in the purchase delivery quantity layout element, a tag element whose text content is "delivery quantity" and the background color is set to gray; adding, in the purchase delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a purchase delivery quantity display tag element for the convenience for the description;
adding, in the purchase data stack layout element, a stack layout element which is referred to as a purchase total delivery quantity stack layout element; adding, in the purchase total delivery quantity layout element, a tag element whose text content is "total delivery quantity" and the background color is set to gray; adding, in the purchase total delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is referred to as a purchase total delivery quantity display tag element for the convenience for the description;
adding, in the purchase instance layout element, a button element which is shortly referred to as the purchase completion button element whose content attribute is set to "purchase completion".

Distributed sales display interaction model

[0049] As shown in Fig. 23, , the construction proce-

dure of the distributed sales display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a root element, a stack layout element referred to as a distributed sales product root layout element; the software interaction mold setting the distributed sales product root layout element to the involved interaction element;
adding, in the distributed sales product root layout element, a tag element as a child interaction element; and setting the text content attribute of the involved tag element to the word "information on the distributed sales";
adding, in the distributed sales product root layout element, a stack layout element which is shortly referred to as the distributed sales contract quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the distributed sales contract quantity stack layout element, a tag element whose text content is "contract quantity" and the background color is set to gray; adding, in the distributed sales contract quantity stack layout element, a textbox element whose text content attribute is set to "0" and which is shortly referred to as the distributed sales contract quantity textbox element;
adding, in the distributed sales product root layout element, a stack layout element which is shortly referred to as the distributed sales shipment quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the distributed sales shipment quantity stack layout element, a tag element whose text content is "shipment quantity" and the background color is set to gray; adding, in the distributed sales shipment quantity stack layout element, a textbox element whose text content attribute is set to "0" and which is shortly referred to as the distributed sales contract quantity textbox element; adding, in the distributed sales shipment quantity stack layout element, a button element whose text content attribute is set to "distributed sales completion" and which is shortly referred to as the distributed sales completion button element.

Direct sales display interaction model

[0050] As shown in Fig. 24, , the construction procedure of the direct sales display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a root element, a stack layout element referred to as a direct sales product root layout element; the software interaction mold setting the direct sales product root layout element to the involved interaction element;
adding, in the direct sales product root layout element, a tag element as a child interaction element; and setting the text content attribute of the involved tag element to the word "information on the direct sales";
adding, in the direct sales product root layout element, a stack layout element which is shortly referred to as the direct sales contract quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the direct sales contract quantity stack layout element, a tag element whose text content is "contract quantity" and the background color is set to gray; adding, in the direct sales contract quantity stack layout element, a textbox element whose text content attribute is set to "0" and which is shortly referred to as the direct sales contract quantity textbox element;
adding, in the direct sales product root layout element, a stack layout element which is shortly referred to as the direct sales shipment quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the direct sales shipment quantity stack layout element, a tag element whose text content is "shipment quantity" and the background color is set to gray; adding, in the direct sales shipment quantity stack layout element, a textbox element whose text content attribute is set to "0" and which is shortly referred to as the direct sales shipment quantity textbox element; adding, in the direct sales shipment quantity stack layout element, a button element whose text content attribute is set to "direct sales completion" and which is shortly referred to as the direct sales completion button element.

Main parts display interaction model

[0051] As shown in Fig. 25, , the construction procedure of the main parts display interaction models similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a root element, a stack layout element referred to as a main parts root layout element; the software interaction mold setting the main parts root layout element to the involved interaction element;
adding, the main parts root layout element, a tag element as a child interaction element; and setting a text content attribute of the involved tag element to the word "information on the main parts";
adding, in the main parts root layout element, a stack layout element which is shortly referred to as the main parts pending processing quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the main parts pending processing quantity stack layout element, a tag element whose text content is "main parts pending processing quantity" and the background color is set to gray; adding, in the main parts pending processing quantity stack layout element, a tag element whose

text content attribute is set to "0" and which is shortly referred to as the main parts pending processing quantity display tag element;

adding, in the main parts root layout element, a stack layout element which is shortly referred to as the main parts processed quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the main parts processed quantity stack layout element, a tag element whose text content is "processed quantity" and the background color is set to gray; adding, in the main parts processed quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the main parts processed quantity display tag element;

adding, in the main parts root layout element, a stack layout element which is shortly referred to as the main parts delivery quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the main parts delivery quantity stack layout element, a tag element whose text content is "delivery quantity" and the background color is set to gray; adding, in the main parts delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the main parts delivery quantity display tag element.

Auxiliary parts display interaction model

[0052] As shown in Fig. 26, , the construction procedure of the auxiliary parts display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a root element, a stack layout element referred to as an auxiliary parts root layout element; the software interaction mold setting the auxiliary parts root layout element as an involved interaction element;

adding, in the auxiliary parts root layout element, a tag element as a child interaction element; and setting a text content attribute of the involved tag element as the word "information on the auxiliary parts";

adding, in the auxiliary parts root layout element, a stack layout element which is shortly referred to as the auxiliary parts pending processing quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the auxiliary parts pending processing quantity stack layout element, a tag element whose text content is "main parts pending processing quantity" and the background color is set to gray; adding, in the auxiliary parts pending processing quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the auxiliary parts pending processing quantity display tag element;

adding, in the auxiliary parts root layout element, a stack layout element which is shortly referred to as the auxiliary parts processed quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the auxiliary parts processed quantity stack layout element, a tag element whose text content is "processed quantity" and the background color is set to gray; adding, in the auxiliary parts processed quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the auxiliary parts processed quantity display tag element;

adding, in the auxiliary parts root layout element, a stack layout element which is shortly referred to as the auxiliary parts delivery quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the auxiliary parts delivery quantity stack layout element, a tag element whose text content is "delivery quantity" and the background color is set to gray; adding, in the auxiliary parts delivery quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the auxiliary parts delivery quantity display tag element.

Finished product display interaction model

[0053] As shown in Fig. 27, , the construction procedure of the finished product display interaction model is similar to that of "business display interaction model" and the content thereof is as follows:

creating, as a root element, a stack layout element referred to as a finished product root layout element; and the software interaction mold setting the finished product root layout element to the involved interaction element;

adding, in the finished product root layout element, a tag element as a child interaction element; setting a text content attribute of the involved tag element to the word "information on the finished product";

adding, in the finished product root layout element, a stack layout element which is shortly referred to as the finished product pending processing quantity stack layout element whose layout direction is set to the horizontal direction; adding, in the finished product pending processing quantity stack layout element, a tag element whose text content is "finished product pending processing quantity" and the background color is set to gray; adding, in the finished product pending processing quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the finished product pending processing quantity display tag element;

adding, in the finished product root layout element, a stack layout element which is shortly referred to as the finished product processed quantity stack layout element whose layout direction is set to the hor-

izontal direction; adding, in the finished product processed quantity stack layout element, a tag element whose text content is "processed quantity" and the background color is set to gray; adding, in the finished product processed quantity stack layout element, a tag element whose text content attribute is set to "0" and which is shortly referred to as the finished product processed quantity display tag element.

Constructing the software algorithm models

[0054] Next, the construction procedure of each software algorithm model will be described in detail.

Product type configuration algorithm model

[0055] Fig. 28 schematically is the accomplished software algorithm model of the product type configuration function of the business management YWGL software component class whose construction procedure is as follows:

the software hierarchy mold receives and responds to the command from the actual software modeling environment to set the business management YWGL software component class to the involved software component class; the software interface mold receives and responds to the command from the actual software modeling environment to set the product type configuration function to the involved function, wherein the software algorithm model for implementing the product type configuration function is shortly referred to as the product type configuration algorithm model in accordance with the function name and software algorithm models for the subsequent other functions may be deduced by analogy, which will not be repeated below;
in the foregoing steps, adding an assignment operator shortly referred to as a produced product type configuration operator which has an input attribute and an output attribute; creating an assignment from the text content attribute of the production product type quantity textbox element in the business display interaction model to the input attribute of produced product type configuration operator; creating an assignment from the output attribute of produced product type configuration operator to the production product type quantity attribute of the business management YWGL software component class;
in the foregoing steps, adding an assignment operator shortly referred to as a purchase product type configuration operator which has an input attribute and an output attribute; creating an assignment from the text content attribute of the purchase product type quantity textbox element in the business display interaction model to the input attribute of the purchase product type configuration operator; creating an assignment from the output attribute of the purchase product type configuration operator to the purchase product type quantity attribute of the business management YWGL software component class;
in the foregoing steps, adding an addition operator shortly referred to as a sales product type configuration operator; creating an assignment from the text content attribute of the production product type quantity textbox element to the augend attribute of the sales product type configuration operator; creating an assignment from the text content attribute of the purchase product type quantity textbox element to the addend attribute of the sales product type configuration operator; creating an assignment from the summation attribute of the sales product type configuration operator to the sales product type quantity attribute of the business management YWGL software component class;

[0056] So far, the product type configuration algorithm model is accomplished.

Sales update algorithm model

[0057] As shown in Fig. 29, , the construction procedure of the sales update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the sales contract quantity update operator which has the following assignments: from the contract quantity attribute of the sales management XSGL software component class to the sales contract quantity update operator; and from the output attribute of the sales contract quantity update operator to the text content attribute of the sales contract quantity display tag element in the sales display interaction model;
an assignment operator which is shortly referred to as the sales receipt quantity update operator which has the following assignments: from the receipt quantity attribute of the sales management XSGL software component class to the input attribute of the sales receipt quantity update operator; and from the output attribute of the sales receipt quantity update operator to the text content attribute of the sales receipt quantity display tag element in the sales display interaction model;
an assignment operator which is shortly referred to as the sales shipment quantity update operator which has the following assignments: from the shipment quantity attribute of the sales management XSGL software component class to the input attribute of the sales shipment quantity update operator; and from the output attribute of the sales shipment quantity update operator to the text content attribute of the sales shipment quantity display tag element in

the sales display interaction model;
an assignment operator which is shortly referred to as the sales total shipment quantity update operator which has the following assignments: from the total shipment quantity attribute of the sales management XSGL software component class to the input attribute of the sales total shipment quantity update operator; and from the output attribute of the sales total shipment quantity update operator to the text content attribute of the sales total shipment quantity display tag element in the sales display interaction model.

Production delivery algorithm model

[0058] As shown in Fig. 30, the construction procedure of the production delivery algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the production completion delivery assignment operator which has the following assignments: from the processed quantity attribute of the production management SCGL software component class to the input attribute of the production completion delivery assignment operator; and from the output attribute of the production completion delivery assignment operator to the delivery quantity attribute of the production management SCGL software component class;
an addition operator which is shortly referred to as the production total delivery quantity operator which has the following assignments: from the processed quantity attribute of the production management SCGL software component class to the augend attribute of the production total delivery quantity operator; from the total delivery quantity attribute of the production management SCGL software component class to the addend attribute of the production total delivery quantity operator; and from the summation attribute of the production total delivery quantity operator to the total delivery quantity attribute of the production management SCGL software component class;
an subtraction operator which is shortly referred to as the production completion reset operator which has the following assignments: from the processed quantity attribute of the production management SCGL software component class to the minuend attribute of the production completion reset operator; from the processed quantity attribute of the production management SCGL software component class to the subtrahend attribute of the production completion reset operator; and from the margin attribute of the production completion reset operator to the processed quantity attribute of the production management SCGL software component class;

Production update algorithm model

[0059] As shown in Fig. 31, the construction procedure of the production update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the production order quantity update operator which has the following assignments: from the order quantity attribute of the production management SCGL software component class to the input attribute of the production order quantity update operator; and from the output attribute of the production order quantity update operator to the text content attribute of the production order quantity display tag element in the production display interaction model;
an assignment operator which is shortly referred to as the production completion quantity update operator which has the following assignments: from the processed quantity attribute of the production management SCGL software component class to the input attribute of the production completion quantity update operator; and from the output attribute of the production completion quantity update operator to the text content attribute of the production completion quantity display tag element in the production display interaction model;
an assignment operator which is shortly referred to as the production delivery quantity update operator which has the following assignments: from the delivery quantity attribute of the production management SCGL software component class to the input attribute of the production delivery quantity update operator; and from the output attribute of the production delivery quantity update operator to the text content attribute of the production delivery quantity display tag element in the production display interaction model;
an assignment operator which is shortly referred to as the production total delivery quantity update operator which has the following assignments: from the total delivery quantity attribute of the production management SCGL software component class to the input attribute of the production total delivery quantity update operator; and from the output attribute of the production total delivery quantity update operator to the text content attribute of the production total delivery quantity display tag element in the production display interaction model.

Purchase implementation algorithm model

[0060] As shown in Fig. 32, the construction procedure of the purchase implementation algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the purchased quantity assignment operator which has the following assignments: from the pending purchase quantity attribute of the purchase management CGGL software component class to the input attribute of the purchased quantity assignment operator; and from the output attribute of the purchased quantity assignment operator to the purchased quantity attribute of the purchase management CGGL software component class;

an assignment operator which is shortly referred to as the purchase delivery quantity assignment operator which has the following assignments: from the pending purchase quantity attribute of the purchase management CGGL software component class to the input attribute of the purchase delivery quantity assignment operator; and from the output attribute of the purchase delivery quantity assignment operator to the delivery quantity attribute of the purchase management CGGL software component class;

an addition operator which is shortly referred to as the purchase total delivery quantity summary operator which has the following assignments: from the pending purchase quantity attribute of the purchase management CGGL software component class to the augend attribute of the purchase total delivery quantity summary operator; from the total delivery quantity attribute of the purchase management CGGL software component class to the addend attribute of the purchase total delivery quantity summary operator; and from the summation attribute of the purchase total delivery quantity summary operator to the total delivery quantity of the purchase management CGGL software component class;

an subtraction operator which is shortly referred to as the pending purchasing quantity reset operator which has the following assignments: from the pending purchase quantity attribute of the purchase management CGGL software component class to the minuend attribute of the pending purchasing quantity reset operator; from the pending purchase quantity attribute of the purchase management CGGL software component class to the subtrahend attribute of the pending purchasing quantity reset operator; and from the output attribute of the pending purchasing quantity reset operator to the pending purchase quantity attribute of the purchase management CGGL software component class.

Purchase update algorithm model

[0061] As shown in Fig. 33, the construction procedure of the purchase update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the pending purchasing quantity update operator which has the following assignments: from the pending purchase quantity attribute of the purchase management CGGL software component class to the input attribute of the pending purchasing quantity update operator; and from the output attribute of the pending purchase quantity update operator to the text content attribute of the pending purchasing quantity display tag element in the purchase display interaction model;

an assignment operator which is shortly referred to as the purchased quantity update operator which has the following assignments: from the purchased quantity attribute of the purchase management CGGL software component class to the input attribute of the purchased quantity update operator; and from the output attribute of the purchased quantity update operator to the text content attribute of the purchased quantity display tag element in the purchase display interaction model;

an assignment operator which is shortly referred to as the purchase delivery quantity update operator which has the following assignments: from the delivery quantity attribute of the purchase management CGGL software component class to the input attribute of the purchase delivery quantity update operator; and from the output attribute of the purchase delivery quantity update operator to the text content attribute of the purchase delivery quantity display tag element in the purchase display interaction model;

an assignment operator which is shortly referred to as the purchase total delivery quantity update operator which has the following assignments: from the total delivery quantity attribute of the purchase management CGGL software component class to the input attribute of the purchase total delivery quantity update operator; and from the output attribute of the purchase total delivery quantity update operator to the text content attribute of the purchase total delivery quantity display tag element in the purchase display interaction model;

Distributed sales update algorithm model

[0062] As shown in Fig. 34, the construction procedure of the distributed sales update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the distributed sales contract quantity recording operator which has the following assignments: from the text content attribute of the distributed sales contract quantity textbox element in the distributed sales display interaction model to the input attribute of the distributed sales contract quantity recording operator; and from the output attribute of the distributed sales contract quantity recording operator to the contract quantity attribute of the distributed sales product

FXP software component class;

an assignment operator which is shortly referred to as the distributed sales shipment quantity recording operator which has the following assignments: from the text content attribute of the distributed sales shipment quantity textbox element in the distributed sales display interaction model to the input attribute of the distributed sales shipment quantity recording operator; and from the output attribute of the distributed sales shipment quantity recording operator to the shipment quantity attribute of the distributed sales product FXP software component class.

Direct sales update algorithm model

[0063] As shown in Fig. 35, the construction procedure of the direct sales update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the direct sales contract quantity recording operator which has the following assignments: from the text content attribute of the direct sales contract quantity textbox element in the direct sales display interaction model to the input attribute of the direct sales contract quantity recording operator; and from the output attribute of the direct sales contract quantity recording operator to the contract quantity attribute of direct sales product ZXP software component class;

an assignment operator which is shortly referred to as the direct sales shipment quantity recording operator which has the following assignments: from the text content attribute of the direct sales shipment quantity textbox element in the direct sales display interaction model to the input attribute of the direct sales shipment quantity recording operator; and from the output attribute of the direct sales shipment quantity recording operator to the shipment quantity attribute of the direct sales product ZXP software component class.

Main parts update algorithm model

[0064] As shown in Fig. 36, the construction procedure of the main parts update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the main parts pending processing quantity update operator which has the following assignments: from the main parts pending processing quantity attribute of the main parts ZJ software component class to the input attribute of the main parts pending processing quantity update operator; and from the output attribute of the main parts pending processing quantity update operator to the text content attribute of the main parts pending processing quantity display tag element in the main parts display interaction model;

an assignment operator which is shortly referred to as the main parts processed quantity update operator which has the following assignments: from the processed quantity attribute of the main parts ZJ software component class to the input attribute of the main parts processed quantity update operator; and from the output attribute of the main parts processed quantity update operator to the text content attribute of the main parts processed quantity display tag element in the main parts display interaction model;

an assignment operator which is shortly referred to as the main parts delivery quantity update operator which has the following assignments: from the delivery quantity attribute of the main parts ZJ software component class to the input attribute of the main parts delivery quantity update operator; and from the output attribute of the main parts delivery quantity update operator to the text content attribute of the main parts delivery quantity display tag element in the main parts display interaction model.

Auxiliary parts update algorithm model

[0065] As shown in Fig. 37, the construction procedure of the auxiliary parts update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the auxiliary parts pending processing quantity update operator which has the following assignments: from the auxiliary parts pending processing quantity attribute of the auxiliary parts LJ software component class to the input attribute of the auxiliary parts pending processing quantity update operator; and from the output attribute of the auxiliary parts pending processing quantity update operator to the text content attribute of the auxiliary parts pending processing quantity display tag element in the auxiliary parts display interaction model;

an assignment operator which is shortly referred to as the auxiliary parts processed quantity update operator which has the following assignments: from the processed quantity attribute of the auxiliary parts LJ software component class to the input attribute of the auxiliary parts processed quantity update operator; and from the output attribute of the auxiliary parts processed quantity update operator to the text content attribute of the auxiliary parts processed quantity display tag element in the auxiliary parts display interaction model;

an assignment operator which is shortly referred to as the auxiliary parts delivery quantity update operator which has the following assignments: from the

delivery quantity attribute of the auxiliary parts LJ software component class to the input attribute of the auxiliary parts delivery quantity update operator; and from the output attribute of the auxiliary parts delivery quantity update operator to the text content attribute of the auxiliary parts delivery quantity display tag element in the auxiliary parts display interaction model.

Finished product update algorithm model

[0066] As shown in Fig. 38, the construction procedure of the finished product update algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the finished product pending processing quantity update operator which has the following assignments: from the finished product pending processing quantity attribute of the finished product CP software component class to the input attribute of the finished product pending processing quantity update operator; and from the output attribute of the finished product pending processing quantity update operator to the text content attribute of the finished product pending processing quantity display tag element in the finished product display interaction model;
an assignment operator which is shortly referred to as the finished product processed quantity update operator which has the following assignments: from the processed quantity attribute of the finished product CP software component class to the input attribute of the finished product processed quantity update operator; and from the output attribute of the finished product processed quantity update operator to the text content attribute of the finished product processed quantity display tag element in the finished product display interaction model;

Main parts processing algorithm model

[0067] As shown in Fig. 39, the construction procedure of the main parts processing algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the main parts pending processing / processed assignment operator which has the following assignments: from the main parts pending processing quantity attribute of the main parts ZJ software component class to the input attribute of the main parts pending processing / processed assignment operator; and from the output attribute of the main parts pending processing / processed assignment operator to the processed quantity attribute of the main parts ZJ software component class;
a subtraction operator which is shortly referred to as the main parts pending processing reset operator which has the following assignments: from the main parts pending processing quantity attribute of the main parts ZJ software component class to the minuend attribute of the main parts pending processing reset operator; from the main parts pending processing quantity attribute of the main parts ZJ software component class to the subtrahend attribute of the main parts pending processing reset operator; and from the margin attribute of the main parts pending processing reset operator to the pending processing quantity attribute of the main parts ZJ software component class.

Main parts delivery algorithm model

[0068] As shown in Fig. 40, the construction procedure of the main parts delivery algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an addition operator which is shortly referred to as the main parts processed delivery operator which has the following assignments: from the processed quantity attribute of the main parts ZJ software component class to the input attribute of the main parts processed delivery operator; and from the output attribute of the main parts processed delivery operator to the delivery quantity attribute of the main parts ZJ software component class;
an addition operator which is shortly referred to as the main parts total delivery quantity operator which has the following assignments: from the processed quantity attribute of the main parts ZJ software component class to the augend attribute of the main parts total delivery quantity operator; from the total delivery quantity attribute of the main parts ZJ software component class to the addend attribute of the main parts total delivery quantity operator; and from the summation attribute of the main parts total delivery quantity operator to the total delivery quantity attribute of the main parts ZJ software component class;
a subtraction operator which is shortly referred to as the main parts processed reset operator which has the following assignments: from the processed quantity attribute of the main parts ZJ software component class to the minuend attribute of the main parts processed reset operator; from the processed quantity attribute of the main parts ZJ software component class to the subtrahend attribute of the main parts processed reset operator; and from the margin attribute of the main parts processed reset operator to the processed quantity attribute of the main parts ZJ software component class.

Auxiliary parts processing algorithm model

[0069] As shown in Fig. 41, the construction procedure of the auxiliary parts processing algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an assignment operator which is shortly referred to as the auxiliary parts pending processing / processed assignment operator which has the following assignments: from the auxiliary parts pending processing quantity attribute of the auxiliary parts LJ software component class to the input attribute of the auxiliary parts pending processing / processed assignment operator; and from the output attribute of the auxiliary parts pending processing / processed assignment operator to the processed quantity attribute of the auxiliary parts ZJ software component class;

a subtraction operator which is shortly referred to as the auxiliary parts pending processing reset operator which has the following assignments: from the auxiliary parts pending processing quantity attribute of the auxiliary parts LJ software component class to the minuend attribute of the auxiliary parts pending processing reset operator; from the auxiliary parts pending processing quantity attribute of the auxiliary parts LJ software component class to the subtrahend attribute of the auxiliary parts pending processing reset operator; and from the margin attribute of the auxiliary parts pending processing reset operator to the auxiliary parts pending processing quantity attribute of the auxiliary parts LJ software component class.

Auxiliary parts delivery algorithm model

[0070] As shown in Fig. 42, the construction procedure of the auxiliary parts delivery algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an addition operator which is shortly referred to as the auxiliary parts processed delivery operator which has the following assignments: from the processed quantity attribute of the auxiliary parts LJ software component class to the input attribute of the auxiliary parts processed delivery operator; and from the output attribute of the auxiliary parts processed delivery operator to the delivery quantity attribute of the auxiliary parts LJ software component class;

an addition operator which is shortly referred to as the auxiliary parts total delivery quantity operator which has the following assignments: from the processed quantity attribute of the auxiliary parts LJ software component class to the augend attribute of the auxiliary parts total delivery quantity operator; from the total delivery quantity attribute of the auxiliary parts LJ software component class to the addend attribute of the auxiliary parts total delivery quantity operator; and from the summation attribute of the auxiliary parts total delivery quantity operator to the total delivery quantity attribute of the auxiliary parts LJ software component class;

a subtraction operator which is shortly referred to as the auxiliary parts processed reset operator which has the following assignments: from the processed quantity attribute of the auxiliary parts LJ software component class to the minuend attribute of the auxiliary parts processed reset operator; from the processed quantity attribute of the auxiliary parts LJ software component class to the subtrahend attribute of the auxiliary parts processed reset operator; and from the margin attribute of the auxiliary parts processed reset operator to the processed quantity attribute of the auxiliary parts LJ software component class.

Auxiliary parts receipt algorithm model

[0071] As shown in Fig. 43, the construction procedure of the auxiliary parts receipt algorithm model is similar to that of "product type configuration algorithm model" and the content thereof is as follows:

an addition operator which is shortly referred to as the main parts receipt operator has the following assignments: from the main parts inventory quantity attribute of the involved software component class to the augend attribute of the main parts receipt operator; from the main parts receipt quantity attribute of the involved software component class to the addend attribute of the main parts receipt operator; and from the summation attribute of the main parts receipt operator to the main parts inventory quantity attribute of the finished product CP software component class;

an addition operator which is shortly referred to as the auxiliary parts receipt operator has the following assignments: from the auxiliary parts inventory quantity attribute of the involved software component class to the augend attribute of the auxiliary parts receipt operator; from the auxiliary parts receipt quantity attribute of the involved software component class to the addend attribute of the auxiliary parts receipt operator; and from the summation attribute of the auxiliary parts receipt operator to the auxiliary parts inventory quantity attribute of the finished product CP software component class.

Finished product assembly algorithm model

[0072] As shown in Fig. 44, the construction procedure of the finished product assembly algorithm model is similar to that of the "product type configuration algorithm model" and the content thereof is as follows:

a multiplication operator which is shortly referred to as the main parts assembly operator has the following assignments: from the pending processing quantity attribute of the involved software component class to the multiplicand input attribute of the main parts assembly operator; and from the single set main parts quantity attribute of the involved software component class to the multiplier input attribute of the main parts assembly operator;

a subtraction operator which is shortly referred to as the main parts assembly inventory operator has the following assignments: from the main parts inventory quantity attribute of the finished product CP software component class to the minuend attribute of the main parts assembly inventory operator; from the product attribute of the main parts assembly inventory operator to the subtrahend attribute of the main parts assembly inventory operator; and from the margin attribute of the main parts assembly inventory operator to the main parts inventory quantity attribute of the finished product CP software component class;

a multiplication operator which is shortly referred to as the an auxiliary-parts assembly operator has the following assignments: from the pending processing quantity attribute of the involved software component class to the multiplicand input attribute of the auxiliary parts assembly operator; and the single set auxiliary parts quantity attribute of the involved software component class to the multiplier input attribute of the auxiliary parts assembly operator;

a subtraction operator which is shortly referred to as the auxiliary parts assembly inventory operator has the following assignments: from the auxiliary parts inventory quantity attribute of the finished product CP software component class to the minuend attribute of the auxiliary parts assembly inventory operator; and from the product attribute of the auxiliary-parts assembly inventory operator to the subtrahend attribute of the auxiliary parts assembly inventory operator; and from the margin attribute of the auxiliary parts assembly inventory operator to the auxiliary parts inventory quantity attribute of the finished product CP software component class;

an assignment operator which is shortly referred to as the finished product processed operator has the following assignments: from the pending processing quantity attribute of the finished product CP software component class to the input attribute of the finished product processed operator; and from the output attribute of the finished product processed operator to the finished product CP software component class' processed quantity attribute;

a subtraction operator which is shortly referred to as the finished product pending processing reset operator has the following assignments: from the finished product CP software component class' pending processing quantity attribute to the minuend attribute of the finished product pending processing operator; from the pending processing quantity attribute of the involved software component class to the subtrahend attribute of the finished product pending processing operator; and from the margin attribute of the finished product pending processing operator to the pending processing quantity attribute of the finished product CP software component class.

Constructing the software process models

[0073] Next, the construction procedure of each software process model will be described in detail.

Main business procedure process model

[0074] Fig. 45 shows a completed main business procedure process model for the business management YWGL software component class. It is constructed as follows:

the software hierarchy mold receives and responds to the command from the actual software modeling environment to set the business management YWGL software component class as the involved software component classes;

the software interface mold receives and responds to the command from the actual software modeling environment to set the main business procedure function as the involved function, wherein, for the convenience of describing the description, the software process model of the main business procedure function is shortly referred to as the main business procedure process model in accordance with the function name and names of process models for the other functions may be deduced by analogy, which will not be repeated below; and wherein the software process mold constructs the main business procedure process model with an attribute process model as a default;

the software process mold first creates a sequential action as the root action for the main business procedure process model, wherein the sequential action is an operator action with sequential execution internal action function, has a start node and an end node, and may sequentially add other actions between the start node and the end node, and wherein, for the convenience of describing the description, the root action is referred to as a main business procedure root action in accordance with the name of the software process model; it should be noted that the software process mold creates a root action for each software process model as a default, of which the name of the root action may be deduced by analogy and will not be repeated below;

the software process mold receives the command from the actual software modeling environment to add an action based on a business display function of the involved software component class; for the

convenience of describing the description, based on the name of function on which the action depends, the action is shortly referred to as the business display action and names for subsequent actions may be deduced by analogy, which will not be repeated below; and the software process mold adds a business display action in the main business procedure root action in response to the foregoing command;
the software process mold receives the command from the actual software modeling environment to add a frame loop action; and the software process mold adds a frame loop action in the main business procedure root action, wherein the frame loop action is an operator action with a frame loop function, there is a sequence of nodes inside the frame loop action, and each node may accommodates another action; and for the convenience of describing the description, the sequence of the nodes of the frame loop action is referred to as a frame loop sequence and the frame loop action is shortly referred to as a main procedure frame loop action;
the software process mold receives and responds to the command from the actual software modeling environment to add a condition action in the frame loop sequence of the main procedure frame loop action, wherein the condition action is shortly referred to as a main procedure condition action and is an operator action with a condition logic function which has two branch action sequences corresponding to the "true" condition and the "false" condition, respectively;
in the foregoing steps, adding a component action, shortly referred to as a business operation action, based on the business operation function of the involved software component class in the branch action sequence of the main procedure condition action corresponding to the "true" condition, and then successionally adding an action, shortly referred to as an operation display and control action, based on the operation display and control function of the involved software component class in the branch action sequence of the main procedure condition action corresponding to the "true" condition, and adding a component action, shortly referred to as a configuration display and control action, based on the configuration display and control function of the involved software component class in the branch action sequence of the main procedure condition action corresponding to the "false" condition.

[0075] So far, the main business procedure process model is accomplished.

Configuration display and control process model

[0076] As shown in Fig. 46, the construction procedure of the configuration display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding an action for the business configuration function; and establishing an event association between the mouse click event of the business configuration button element in the business display interaction model and the business configuration action.

Business configuration process model

[0077] As shown in Fig. 47, the construction procedure of the business configuration process model is similar to that for the "main business procedure process model" and the content thereof is as follows:

adding, in the business configuration root action, the command based on the component action for the product type configuration function of the involved software component class and adding, in the business configuration root action, a product type configuration action which is shortly referred to as the business product type configuration action; adding, in the business configuration root action, the command based on the component action for the instance creation function of the involved software component class and adding, in the business configuration root action, an instance creation action which is shortly referred to as the business instance creation action; adding, in the business configuration root action, the command based on the component action for the instance configuration function of the involved software component class and adding, in the business configuration root action, an instance configuration action which is shortly referred to as the business instance configuration action; adding, in the business configuration root action, a negating operator action which is shortly referred to as the business operation state negating action.

Business instance creation process model

[0078] As shown in Fig. 48, the construction procedure of the business instance creation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the instance creation root action, an instance creation operator action which is shortly referred to as the sales instance creation action, wherein the instance creation operator action is an operator action with a function for creating the software component instance and has a component class attribute and an instance quantity attribute; adding, in the instance creation root action, an instance creation operator action which is shortly referred to as the production instance creation action; adding, in the instance creation root action, an instance creation operator action which is shortly re-

ferred to as the purchase instance creation action.

Business instance configuration process model

[0079] As shown in Fig. 49, the construction procedure of the business instance configuration process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the business configuration root action, a traversal action is shortly referred to as a production configuration traversal action for the convenience of describing the description, wherein the traversal action is an operator action which is traversally performed on all instances of a specified software component class, and wherein, inside the traversal action, there is a sequence of nodes each of which accommodates an action;
adding, in the sequence of nodes inside the production configuration traversal action, an increment action which is shortly referred to as the production serial number increment action for the convenience of describing the description, wherein the increment action is a prefabricated operator action with a function for allowing an integer to increase by one; adding, in the sequence of nodes inside the production configuration traversal action, an assignment action which is shortly referred to as the production serial number assignment action;
adding, in the business configuration root action, a traversal action which is shortly referred to as the purchase configuration traversal action;
adding, in the sequence of nodes inside the purchase configuration traversal action, an increment action which is shortly referred to as the purchase serial number increment action; adding, in the sequence of nodes inside the purchase configuration traversal action, an assignment action which is shortly referred to as the purchase serial number assignment action;
adding, in the business configuration root action, an assignment action which is shortly referred to as the sales serial number reset action; adding, in the business configuration root action, a traversal action which is shortly referred to as the sales configuration traversal action;
adding, in the sequence of nodes inside the sales configuration traversal action, an increment action which is shortly referred to as the sales serial number increment action; adding, in the sequence of nodes inside the sales configuration traversal action, an assignment action which is shortly referred to as the sales serial number assignment action; adding, in the sequence of nodes inside the sales configuration traversal action, a traversal action which is shortly referred to as the sales-production configuration traversal action;
adding, in the sequence of nodes inside the sales-production configuration traversal action, a consistency comparison action which is shortly referred to as the sales-production configuration comparison action and which is an operator action with a decision function for comparing whether or not two inputs are consistent; adding, in the sequence of nodes inside the sales-production configuration traversal action, a condition action which is shortly referred to as the sales-production configuration condition action, which is an operator action with a condition logic function;
adding, in the "true" branch of the sales-production configuration condition action, an assignment action which is shortly referred to as the sales-production product name assignment action;
adding, in the sequence of nodes inside the sales configuration traversal action, a traversal action which is shortly referred to as the sales-purchase configuration traversal action;
adding, in the sequence of nodes inside the sales-purchase configuration traversal action, a consistency comparison action which is shortly referred to as the sales-purchase configuration comparison action; adding, in the sequence of nodes inside the sales-purchase configuration traversal action, a condition action which is shortly referred to as the sales-purchase configuration condition action;
adding, in the "true" branch of the sales-purchase configuration condition action, an assignment action which is shortly referred to as the sales-purchase product name assignment action.

Business operation process model

[0080] As shown in Fig. 50, the construction procedure of the business operation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the business operation root action, a traversal action which is shortly referred to as the sales operation traversal action;
adding, in the sequence of nodes inside the sales operation traversal action, a traversal action which is shortly referred to as the sales-production operation traversal action;
adding, in the sequence of nodes inside the sales-production operation traversal action, a consistency comparison action which is shortly referred to as the sales-production operation comparison action; adding, in the sequence of nodes inside the sales-production operation traversal action, a condition action which is shortly referred to as the sales-production operation condition action;
adding, in the "true" branch of the sales-production operation traversal action, an action based on the sales operation function of the sales management XSGL software component class, shortly referred to

as a sales-production operation action;

adding, in the sequence of nodes inside the sales operation traversal action, a traversal action which is shortly referred to as the sales-purchase operation traversal action;

adding, in the sequence of nodes inside the sales-purchase operation traversal action, a consistency comparison action which is shortly referred to as the sales-purchase operation comparison action; adding, in the sequence of nodes inside the sales-purchase operation traversal action, a condition action which is shortly referred to as the sales-purchase operation condition action;

adding, in the "true" branch of the sales-purchase operation traversal action, an action based on the sales operation function of the sales management XSGL software component class, shortly referred to as a sales-purchase operation action.

Operation display and control process model

[0081] As shown in Fig. 51, the construction procedure of the operation display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the operation display and control root action, a traversal action which is shortly referred to as the sales display and control traversal action;

adding, in the sequence of nodes in the sales display and control traversal action, an action based on the sales display and control function of the sales management XSGL software component class, shortly referred to as a sales display and control action;

adding, in the operation display and control root action, a traversal action which is shortly referred to as the production display and control traversal action;

adding, in the sequence of nodes in the production display and control traversal action, an action based on the production display and control function of the production management SCGL software component class, shortly referred to as a production display and control action;

adding, in the operation display and control root action, a traversal action which is shortly referred to as the purchase display and control traversal action;

adding, in the sequence of nodes in the purchase display and control traversal action, a component action based on the purchase display and control function of the purchase management CGGL software component class, shortly referred to as a purchase display and control action.

Sales display and control process model

[0082] As shown in Fig. 52, the construction procedure of the sales display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the sales display and control root action, a component action based on the distributed sales display and control function of the distributed sales product FXP software component class, shortly referred to as a distributed sales display and control action; adding, in the sales display and control root action, a component action based on the direct sales display and control function of the direct sales product ZXP software component class, shortly referred to as a direct sales display and control action; adding, in the sales display and control root action, a component action based on the sales update function of the involved software component class, shortly referred to as a sales update action.

Sales operation process model

[0083] As shown in Fig. 53, the construction procedure of the sales operation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the sales operation root action, an addition action which is shortly referred to as the sales receipt action and which is an operator action with an addition function; adding, in the sales operation root action, an addition action which is shortly referred to as the shipment quantity summary action; adding, in the sales operation root action, an addition action which is shortly referred to as the total shipment quantity summary action; adding, in the sales operation root action, a subtraction action which is shortly referred to as the inventory quantity summary action; adding, in the sales operation root action, an addition action which is shortly referred to as the contract quantity summary action; adding, in the sales operation root action, an addition action which is shortly referred to as the demand quantity summary action; adding, in the sales operation root action, a subtraction action which is shortly referred to as the order quantity summary action.

Production display and control process model

[0084] As shown in Fig. 54, the construction procedure of the production display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding an action based on the production operation function of the involved software component class, shortly referred to as a production operation action;

establishing an event association between the mouse click event of the production completion button element in the production display interaction model and the production operation action.

production operation process model

[0085] As shown in Fig. 55, the construction procedure of the production operation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the production operation root action, an action based on the production planning function of the involved software component class, shortly referred to as a production planning action; adding, in the production operation root action, an action based on the production implementation function of the involved software component class, shortly referred to as a production implementation action; adding, in the production operation root action, an action based on the production delivery function of the involved software component class, shortly referred to as a production delivery action; adding, in the production operation root action, an action based on the production update function of the involved software component class, shortly referred to as a production update action; adding, in the production operation root action, an action based on the main parts update function of the main parts ZJ software component class, shortly referred to as a main parts update action; adding, in the production operation root action, an action based on the auxiliary parts update function of the auxiliary parts LJ software component class, shortly referred to as an auxiliary parts update action; adding, in the production operation root action, an action based on the finished product update function of the finished product CP software component class, shortly referred to as a finished product update action.

Production planning process model

[0086] As shown in Fig. 56, the construction procedure of the production planning process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the production planning root action, a multiplication action which is shortly referred to as the main parts pending processing quantity summary action and which is an operator action based on the multiplication function of the multiplication operator; adding, in the production planning root action, another multiplication action which is shortly referred to as the auxiliary parts pending processing quantity summary action.

Production implementation process model

[0087] As shown in Fig. 57, the construction procedure of the production implementation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the production implementation root action, an action based on the main parts processing function of the main parts ZJ software component class, shortly referred to as a main parts processing action; adding, in the production implementation root action, an action based on the main parts delivery function of the main parts ZJ software component class, shortly referred to as a main parts delivery action; adding, in the production implementation root action, an action based on the auxiliary parts processing function of the auxiliary parts LJ software component class, shortly referred to as an auxiliary parts processing action; adding, in the production implementation root action, an action based on the auxiliary parts delivery function of the auxiliary parts LJ software component class, shortly referred to as an auxiliary parts delivery action; adding, in the production implementation root action, an action based on the auxiliary parts receipt function of the finished product CP software component class, shortly referred to as an auxiliary parts receipt action; adding, in the production implementation root action, an action based on the finished product assembly function of the finished product CP software component class, shortly referred to as a finished product assembly action.

Purchase display and control process model

[0088] As shown in Fig. 58, the construction procedure of the purchase display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding an action based on the purchase operation function of the involved software component class, shortly referred to as a purchase operation action; establishing an event association between the mouse click event of the purchase completion button element in the purchase display interaction model and the purchase operation action.

Purchase operation process model

[0089] As shown in Fig. 59, the construction procedure of the purchase operation process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding, in the purchase operation root action, an action based on the purchase implementation function of the purchase management CGGL software component class, shortly referred to as a purchase implementation action; adding, in the purchase operation root action, an action based on the purchase update function of the purchase management CGGL

software component class, shortly referred to as a purchase update action.

Distributed sales display and control process model

[0090] As shown in Fig. 60, the construction procedure of the distributed sales display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding an action based on the distributed sales update function of the involved software component class, shortly referred to as a distributed sales update action; establishing an event association between the mouse click event of the distributed sales completion button element in the distributed sales display interaction model and the distributed sales update action.

Direct sales display and control process model

[0091] As shown in Fig. 61, the construction procedure of the direct sales display and control process model is similar to that of "main business procedure process model" and the content thereof is as follows:

adding an action based on the direct sales update function of the involved software component class, shortly referred to as a direct sales update action; establishing an event association between the mouse click event of the direct sales completion button element in the direct sales display interaction model and the direct sales update action.

Constructing the software transfer models

[0092] Next, the construction procedure of the software transfer model for each action will be described in detail.

Business display transfer model

[0093] (null)

Main procedure frame loop transfer model

[0094] Fig. 62 shows a completed main procedure frame loop transfer model whose construction procedure is as follows:

the software hierarchy mold receives and responds to the command from the actual software modeling environment to set the business management YWGL software component class as the involved software component class;
the software interface mold receives and responds to the command from the actual software modeling environment to set the main business procedure function as the involved function;
the software process mold receives command from the actual software modeling environment to set the main procedure frame loop action as the involved action; the software transfer mold constructs a software transfer model for the involved action; for simplicity, the software transfer model for main procedure frame loop action is shortly referred to as the main procedure frame loop transfer model in accordance with name of the involved action; names of software transfer models for other actions may be deduced by analogy, which will not be repeated; and
the software transfer mold receives the command from the actual software modeling environment to establish an input transfer from the main loop state attribute of the involved software component class to the state attribute of the involved action; the software transfer mold establishes an input transfer from the main loop state attribute of the business management YWGL software component class to the state attribute of the main procedure frame loop action in response to the foregoing command, wherein the state attribute of the main procedure frame loop action, as a Boolean variable, refers to an abbreviation of a state attribute for the business loop operation action to control whether or not operates; and
names of the subsequent action's attributes may be deduced by analogy, which will not be repeated.

[0095] So far, the main procedure frame loop transfer model is accomplished.

Main procedure condition transfer model

[0096] Fig. 63 is a completed main procedure condition transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the business operation state attribute of the business management YWGL software component class to the state attribute of the main procedure condition action.

Configuration display and control transfer model

[0097] (null)

Business operation transfer model

[0098] (null)

Operation display and control transfer model

[0099] (null)

Business configuration transfer model

[0100] (null)

Business type configuration transfer model

[0101] (null)

Business instances creation transfer model

[0102] (null)

Business instance configuration transfer model

[0103] (null)

Business operation state negating transfer model

[0104] Fig. 64 is a completed business operation state negating transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the business operation state attribute of the involved software component class to the input of the operation state negating action; and from the output of the operation state negating action to the business operation state attribute of the involved software component class.

Sales instance creation transfer model

[0105] Fig. 65 is a completed sales instance creation transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the name of the sales management XSGL software component class to the type attribute of the sales instance creation action; and from the sales product type quantity attribute of the involved software component class to the instance quantity attribute of the sales instance creation action.

Production instance creation transfer model

[0106] Fig. 66 is a completed production instance creation transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the name of the production management SCGL software component class to the Component class attribute of the production instance creation action; and from the production product type quantity attribute of the involved software component class to the instance quantity attribute of the production instance creation action.

Purchase instance creation transfer model

[0107] Fig. 67 is a completed purchase instance creation transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the name of the purchase management CGGL software component class to the Component class attribute of the purchase instance creation action; and from the purchase product type quantity attribute of the involved software component class to the instance quantity attribute of the purchase instance creation action.

Production configuration traversal transfer model

[0108] Fig. 68 is a completed production configuration traversal transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the name of the production management SCGL software component class to the type attribute of the production configuration traversal action.

Production serial number increment transfer model

[0109] Fig. 69 is a completed production serial number increment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the product serial number attribute of the business assistant YWZS software component class to the input attribute of the production serial number increment action; and from the output attribute of the production serial number increment action to the product serial number attribute of the business assistant YWZS software component class.

Production serial number assignment transfer model

[0110] Fig. 70 is a completed production serial number assignment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the product serial number attribute of the business assistant YWZS software component class to

the input attribute of the production serial number assignment action; and from the output attribute of the production serial number assignment action to the product serial number attribute of the production management SCGL software component class.

Purchase configuration traversal transfer model

[0111] Fig. 71 is a completed purchase configuration traversal transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the name of the purchase management CGGL software component class to the type attribute of the purchase configuration traversal action.

Purchase serial number increment transfer model

[0112] It is similar to the production serial number increment transfer model.

Purchase serial number assignment transfer model

[0113] Fig. 72 is a completed purchase serial number assignment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the product serial number attribute of the business assistant YWZS software component class to the input attribute of the purchase serial number assignment action; and from the output attribute of the purchase serial number assignment action to the product serial number attribute of the purchase management SCGL software component class.

Sales serial number reset transfer model

[0114] Fig. 73 is a completed sales serial number reset transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the constant zero attribute of the business assistant YWZS software component class to the input attribute of the sales serial number reset action; and from the output attribute of the sales serial number reset action to the product serial number attribute of the business assistant YWZS software component class.

Sales configuration traversal transfer model

[0115] Fig. 74 is a completed sales configuration traversal transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the name of the sales management XSGL software component class to the type attribute of the sales configuration traversal action.

Sales serial number increment transfer model

[0116] It is similar to the production serial number increment transfer model.

Sales serial number assignment transfer model

[0117] Fig. 75 is a completed sales serial number assignment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the product serial number attribute of the business assistant YWZS software component class to the input attribute of the sales serial number assignment action; and from the output attribute of the sales serial number assignment action to the product serial number attribute of the sales management XSGL software component class.

Sales-production configuration traversal transfer model

[0118] It is similar to the production configuration traversal transfer model.

Sales-production configuration comparison transfer model

[0119] Fig. 76 shows a completed sales-production configuration comparison transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the sales management XSGL software component class' product serial number attribute to sales-production configuration comparison action's comparison attribute; from the production management SCGL software component class' product serial number attribute to sales-production configuration comparison action's comparison attribute; and from the sales-production configuration comparison action's result attribute to the business assistant YWZS software component class' comparison result attribute.

Sales-production configuration condition transfer model

[0120] Fig. 77 shows a completed sales-production configuration condition transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the business assistant YWZS software component class' comparison result attribute to sales-production configuration condition action's state attribute.

Sales-production product name assignment transfer model

[0121] Fig. 78 shows a completed sales-production product name assignment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the production management SCGL software component class' product name attribute to sales-production product name assignment action's input attribute; and from the sales-production product name assignment action's output attribute to sales management XSGL software component class' product name attribute.

Sales-purchase configuration traversal transfer model

[0122] It is similar to the purchase configuration traversal transfer model.

Sales-purchase configuration comparison transfer model

[0123] Fig. 79 shows a completed sales-purchase configuration comparison transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the sales management XSGL software component class' product serial number attribute to the sales-purchase configuration comparison action's comparison attribute; from the purchase management CGGL software component class' product serial number attribute to sales-purchase configuration comparison action's comparison attribute; and from the sales-purchase configuration comparison action's result attribute to business assistant YWZS software component class' comparison result attribute.

Sales-purchase configuration condition transfer model

[0124] Fig. 80 shows a completed sales-purchase configuration condition transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the business assistant YWZS software component class' comparison result attribute to the sales-purchase configuration condition action's state attribute.

Sales-purchase product name assignment transfer model

[0125] Fig. 81 shows a completed sales-purchase product name assignment transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the purchase management CGGL software component class' product name attribute to sales-purchase product name assignment action's input attribute; and from sales-purchase product name assignment action's output attribute to sales management XSGL software component class' product name attribute.

Sales operation traversal transfer model

[0126] It is similar to the sales configuration traversal transfer model.

Sales-production operation traversal transfer model

[0127] It is similar to the sales-production configuration traversal transfer model.

Sales-production operation comparison transfer model

[0128] It is similar to the sales-production configuration comparison transfer model.

Sales-production operation condition transfer model

[0129] It is similar to the sales-production configuration condition transfer model.

Sales-production operation transfer model

[0130] Fig. 82 shows a completed sales-production operation transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer

model" and the content thereof contains the following transfers:

from the production management SCGL software component class' delivery quantity attribute to sales-production operation action's receipt quantity attribute; and from the sales-production operation action's order quantity attribute to production management SCGL software component class' order quantity attribute.

Sales-purchase operation traversal transfer model

[0131] It is similar to the sales-purchase configuration traversal transfer model.

Sales-purchase operation comparison transfer model

[0132] It is similar to the sales-purchase configuration comparison transfer model.

Sales-purchase operation condition transfer model

[0133] It is similar to the sales-purchase configuration condition transfer model.

Sales-purchase operation transfer model

[0134] Fig. 83 shows a completed sales-purchase operation transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the purchase management CGGL software component class' delivery quantity attribute to the sales-purchase operation action's receipt quantity attribute; and from the sales-purchase operation action's order quantity attribute to sales management XSGL software component class' pending purchase quantity attribute.

Sales receipt transfer model

[0135] Fig. 84 shows a completed sales receipt transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the inventory quantity attribute of the sales management XSGL software component class to the augend attribute of the sales receipt action; from the receipt quantity attribute of the sales management XSGL software component class to the addend attribute of the sales receipt action, and from the summation attribute of the sales receipt action to the inventory quantity attribute of the sales management XSGL software component class.

Shipment quantity summary transfer model

[0136] Fig. 85 is a completed shipment quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the shipment quantity attribute of the distributed sales product FXP software component class to the augend attribute of the shipment quantity summary action; from the shipment quantity attribute of the direct sales product ZXP software component class to the addend attribute of the shipment quantity summary action; and from the summation attribute of the shipment quantity summary action to the shipment quantity attribute of the sales management XSGL software component class.

Total shipment quantity summary transfer model

[0137] Fig. 86 is a completed total shipment quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the shipment quantity attribute of the sales management XSGL component class to the augend attribute of the total shipment quantity summary action; from the total shipment quantity attribute of the sales management XSGL software component class to the addend attribute of the total shipment quantity summary action; and from the summation attribute of the total shipment quantity summary action to the total shipment quantity attribute of the sales management XSGL software component class.

Inventory quantity summary transfer model

[0138] Fig. 87 is a completed inventory quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the inventory quantity attribute of the sales management XSGL component class to the minuend attribute of the inventory quantity summary action; from the shipment quantity attribute of the sales management XSGL software component class to the subtrahend attribute of the inventory quantity summary action, and from the margin attribute of the inventory quantity summary action to the inventory quantity attribute of the sales management XSGL software component class.

Contract quantity summary transfer model

[0139] Fig. 88 is a completed contract quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the contract quantity attribute of the distributed sales product FXP software component class to the augend attribute of the contract quantity summary action; from the contract quantity attribute of the direct sales product ZXP software component class to the addend attribute of the contract quantity summary action, and from the summation attribute of the contract quantity summary action to the contract quantity attribute of the sales management XSGL software component class.

Demand quantity summary transfer model

[0140] Fig. 89 is a completed demand quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the contract quantity attribute of the sales management XSGL component class to the augend attribute of the demand quantity summary action; from the minimum inventory attribute of the sales management XSGL software component class to the addend attribute of the demand quantity summary action, and from the summation attribute of the demand quantity summary action to the demand quantity attribute of the sales management XSGL software component class.

Order quantity summary transfer model

[0141] Fig. 90 is a completed order quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the demand quantity attribute of the sales management XSGL component class to the minuend attribute of the order quantity summary action; from the inventory quantity attribute of the sales management XSGL software component class to the subtrahend attribute of the order quantity summary action, and from the margin attribute of the order quantity summary action to the order quantity attribute of the sales management XSGL software component class.

Sales display and control transfer model

[0142] (null)

Distributed sales display and control transfer model

[0143] (null)

Direct sales display and control transfer model

[0144] (null)

Sales update transfer model

[0145] (null)

Production display and control traversal transfer model

[0146] It is similar to the production configuration traversal transfer model.

Production display and control transfer model

[0147] (null)

Production operation transfer model

[0148] (null)

Production planning transfer model

[0149] (null)

Production implementation transfer model

[0150] (null)

Production delivery transfer model

[0151] (null)

Production update transfer model

[0152] (null)

Main parts pending processing quantity summary transfer model

[0153] Fig. 91 is a completed main parts pending processing quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the order quantity attribute of the production management SCGL software component class to the multiplicand attribute of the main parts pending

processing quantity summary action; from the single set main parts quantity attribute of the finished product CP software component class to the multiplier attribute of the main parts pending processing quantity summary action; and from the product attribute of the main parts pending processing quantity summary action to the pending processing quantity attribute of the main parts ZJ software component class.

Auxiliary parts pending processing quantity summary transfer model

[0154] Fig. 92 is a completed auxiliary parts pending processing quantity summary transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the order quantity attribute of the production management SCGL software component class to the multiplicand attribute of the auxiliary parts pending processing quantity summary action; from the single set auxiliary parts quantity attribute of the finished product CP software component class to the multiplier attribute of the auxiliary parts pending processing quantity summary action; and from the product attribute of the auxiliary parts pending processing quantity summary action to the pending processing quantity attribute of the auxiliary parts LJ software component class.

Main parts processing transfer model

[0155] (null)

Main parts delivery transfer model

[0156] (null)

Auxiliary parts processing transfer model

[0157] (null)

Auxiliary parts delivery transfer model

[0158] (null)

Parts receipt transfer model

[0159] Fig. 93 is a completed parts receipt transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfers:

from the delivery quantity attribute of the main parts ZJ software component class to the main parts receipt quantity attribute of the parts receipt action; and from the delivery quantity attribute of the auxiliary parts LJ software component class to the auxiliary parts receipt quantity attribute of the parts receipt action.

Finished product assembly transfer model

[0160] Fig. 94 is a completed finished product assembly transfer model whose construction procedure is similar to that of the "main procedure frame loop transfer model" and the content thereof contains the following transfer:

from the processed quantity attribute of the finished product assembly action to the processed quantity attribute of the production management SCGL software component class.

Main parts update transfer model

[0161] (null)

auxiliary parts update transfer model

[0162] (null)

Finished product update transfer model

[0163] (null)

Purchase display and control transfer model

[0164] (null)

Purchase operation transfer model

[0165] (null)

Purchase implementation transfer model

[0166] (null)

Purchase update transfer model

[0167] (null)

[0168] Thereby, the business management YWGL software model constituted by a software hierarchy model, software interaction models, software interface models, software algorithm models, software process models, and software transfer models in this embodiment has been accomplished.

[0169] This embodiment demonstrates how a regular management personnel, without knowledge of any existing complex software modeling languages, without knowledge of any computer programming language, and without dependence on any professional modeler nor any application developer, by using the present invention, independently constructs an executable business man-

agement software model based on his vision in business management within a relatively short period of time. The constructed software model is not only clear and simple but also the quality of the constructed software model is significantly higher and the time spent is significantly shorter.

[0170] Compared with developing a business management software model with the cooperation of professional modelers and/or application developers, the present invention by which the same manager independently develops the business management software model, achieves remarkable results as follows:

(1) higher quality: the completed software model meets the minds of the managers and avoids the possible bias in understanding of the business management software model between the managers and professional modelers or application developers;
(2) shorter time spent: the entire period of time spent to model is shortened to 1/5 of the original time period because the complex and frequent communications between the managers and the professional modelers or the application developers are eliminated, thereby greatly saving energy and money.

Claims

1. Software element model based univeral software modeling method to construct software model, by means of a computer readable storage medium having a computer readable program code stored therein, the computer readable program code containing instructions executable by a processor of a computer software to implement a method of constructing software model by processing data conforming to the software element model and describing the software model, the software model describing a software system, the software element model comprising:

a software hierarchy mold which describes the software hierarchy model of the software model in a tree structure whose nodes are software component classes and which is used as a template to be configured in an actual software modeling environment to form the software hierarchy model of the software model, wherein the software hierarchy model refers to the hierarchy relationships constituted by the software component classes as the nodes in the software model, wherein the software component class refers to a set of software component instances with the same external features, and wherein the tree structure, whose nodes are the software component classes, is referred as a hierarchy tree;
a software interface mold which describes software interface models by an optional structure of an attribute set, a function set, and an event set, the software interface mold is used as a template in the actual software modeling environment to be configured to form the software interface models, wherein the software interface models refer to external features of the software component classes, wherein the functions in the function set include software interaction functions, software algorithm functions, and software process functions, wherein the software interaction function is implemented by a software interaction model, wherein the software algorithm function is implemented by a software algorithm model, and wherein the software process function is implemented by a combination of software process models and software transfer models;
a software interaction module which describes the software interaction models by a tree structure whose nodes are interaction elements and which is used as a template in the actual software modeling environment to be configured to form the software interaction models, wherein the software interaction model refers to a description of a way for implementing the software interaction function with a combination of the interaction elements and the interaction element refers to a functional element for interacting information with the actual software modeling environment;
a software algorithm mold which describes software algorithm models by a tree structure whose nodes are operators and which is used as a template in the actual software modeling environment to be configured to form the software algorithm model, wherein the software algorithm model refers to a description of the algorithm which implements the software algorithm function by using the combination of operators, and wherein the operator refers to a component with a previously realized specific function;
a software process mold which describes software process models by combining actions as nodes and which is used as a template in the actual software modeling environment to configure the software process models, wherein the software process model refers to a description of a way of the software process function which is implemented using a combination of the actions and wherein the action refers to an execution of a function;
a software transfer mold which describes software transfer models by a transfer set and which is used as a template in the actual software modeling environment to be configured to form the software transfer models, wherein the software transfer model refers to transfer relationships of the data of involved actions and a transfer in the transfer set is a transfer relationship of the data

between one attribute and another attribute;
specific steps to construct the software model described by the six molds being as follows:

1) constructing the software hierarchy model: the software hierarchy mold reading in software hierarchy model commands from the actual software modeling environment, wherein the software hierarchy model command refers to command such as creating a software component class, adding a software component class, selecting a software component class, naming a software component class, or deleting a software component class for the hierarchy tree and wherein the software hierarchy mold performs corresponding operations on the software component class nodes in response to the software hierarchy model commands to obtain the software hierarchy model;
2) constructing the software interface models: constructing the software interface model for each software component class of the software hierarchy model obtained in the step 1), the steps for constructing each software interface model including: the software interface mold reading in software interface model commands from the actual software modeling environment, wherein the software interface model command refers to command such as creating, naming, or deleting the attributes, the functions, and the events, wherein the software interface mold performs corresponding operations in response to the software interface model commands to obtain the software interface model, and wherein the software interaction models for implementing software interaction functions are constructed by step 3), wherein the software algorithm models for implementing software algorithm functions are constructed by step 4), and wherein the software process models for implementing software process functions are constructed by the step 5);
3) constructing the software interactive models: constructing the software interaction model for each software interaction function obtained in the step 2), steps for constructing each software interaction model including: the software interaction mold reading in software interaction model commands from the actual software modeling environment;
4) constructing the software algorithm models: constructing the software algorithm model for each software algorithm function obtained in the step 2), the steps for constructing each software algorithm model including: the software algorithm mold reading in software algorithm model commands from the actual software modeling environment;
5) constructing the software process models: constructing the software process model for each software process function obtained in the step 2), the steps for constructing each software process model including: the software process mold reading in software process model commands from the actual software modeling environment; and
6) constructing the software transfer models: constructing the software transfer model for each action in the software process models obtained in the step 5), the steps for constructing each software transfer model including: the software transfer mold reading in software transfer model commands from the actual software modeling environment, wherein the software transfer model command refers to the command such as adding a transfer, adding a transfer, or deleting a transfer and wherein the software transfer mold performs corresponding operations in response to the software transfer model commands to obtain the software transfer model,
thereby the software model constructed by the software hierarchy model, the software interface models, the software algorithm models, the software process models, and the software transfer models is accomplished.

2. Software element model based univeral software modeling method to construct software model in claim 1, wherein a combination of the software process mold and the software transfer mold provide a universal means to describe and configure functions; the software interaction mold provides a simplified alternative for replacing the combination of the software process mold and the software transfer mold if only interaction elements are used to implement the functions.

3. Software element model based univeral software modeling method to construct software model in claim 1, wherein a combination of the software process mold and the software transfer mold provide a universal means to describe and configure functions; the software algorithm mold provides a simplified alternative for replacing the combination of the software process mold and the software transfer mold if only operators are used to implement the functions.

4. Software element model based univeral software

modeling method to construct software model in claim 1, wherein the software element model employs a parent-child structure as a base recursive unit to recursively describe the software model; the parent-child structure refers to a structure of parent-child relationships in a hierarchy tree, constituted by an involved software component class and all child software component classes thereof.

5. Software element model based univeral software modeling method to construct software model in claim 1, wherein the specific function of the step 2) can only be any one of the software interaction function, the software algorithm function, and software process function.

6. Software element model based univeral software modeling method to construct software model in claim 1, wherein the software interaction model commands for constructing the software interaction model in the step 3) refer to commands, such as adding an interaction element, selecting an interaction element, naming an interaction element, and deleting an interaction element, and the software interaction mold performs corresponding operations in response to the software interaction model commands to obtain the software interaction model; the interaction elements include operator interaction elements and component interaction elements; the operator interaction element refers to a component with a previously realized specific function and the component interaction element refers to one execution of the interaction function in a set of the interaction functions in the parent-child structure, the set of the interaction functions in the parent-child structure refers to a collection constituted by all interaction functions of the involved component class and all interaction functions of all child component classes thereof in the parent-child structure, the tree structure of which the nodes are the interaction elements is referred to as an interaction tree.

7. Software element model based univeral software modeling method to construct software model in claim 1, wherein the software algorithm model commands for constructing the software algorithm model in the step 4) refer to commands, such as adding an operator, selecting an operator, naming an operator, and deleting an operator, as well as adding an assignment, selecting an assignment, and deleting an assignment, and the software algorithm mold performs corresponding operations in response to the software algorithm model commands to obtain the software algorithm model; the operators include logic operators with logic functions and computation operators with calculation functions; the tree structure whose nodes are operators is referred to as an algorithm tree; the assignment refers to an assignment relationship between two attributes in a set of the algorithm attributes; and the set of the algorithm attributes refers to a collection constituted by a set of attributes of the involved software component classes, a set of attributes of all operators, and a set of attributes of all interaction elements in the software interaction model.

8. Software element model based univeral software modeling method to construct software model in claim 1, wherein the software process model commands for constructing the software process model in step 5) refer to commands such as adding an action, selecting an action, naming an action, and deleting an action and the software process mold performs a corresponding operation in response to the software process model commands to obtain the software process model; the actions include both component actions and operator actions; the component action refers to one execution of the functions in the function set in the parent-child structure, the function set in the parent-child structure refers to a collection constituted by the function set of the involved software component class and function sets of all child software component classes in the parent-child structure; the operator action refers to one execution of operator's function; the software process models include attribute process models and event process model, the software process mold includes attribute process molds and event process mold, and the attribute process mold describes an attribute process model through a process tree as the structure, which is a tree structure constituted by actions as nodes; the event process mold describes an event process model by a set of event associations as the structure; the event association in the set of event associations is an association relationship between an event in a set of events in a parent-child structure and an operator action or a component action; the event set in the parent-child structure is a collection constituted by the event set of the involved software component class and the event sets of all interaction elements in the interaction model thereof and the event sets of all child software component classes and the event sets of all interaction elements in the interaction model thereof, in the parent-child structure.

9. Software element model based univeral software modeling method to construct software model in claim 1, wherein, besides action attributes which refers to the attribute of the component class where the action is, the attributes relevant to transfers are limited to the parent-child attribute set, which refers to a collection constituted by the attribute set of the involved software component classes and attribute sets of all child software component classes thereof in the parent-child structure.

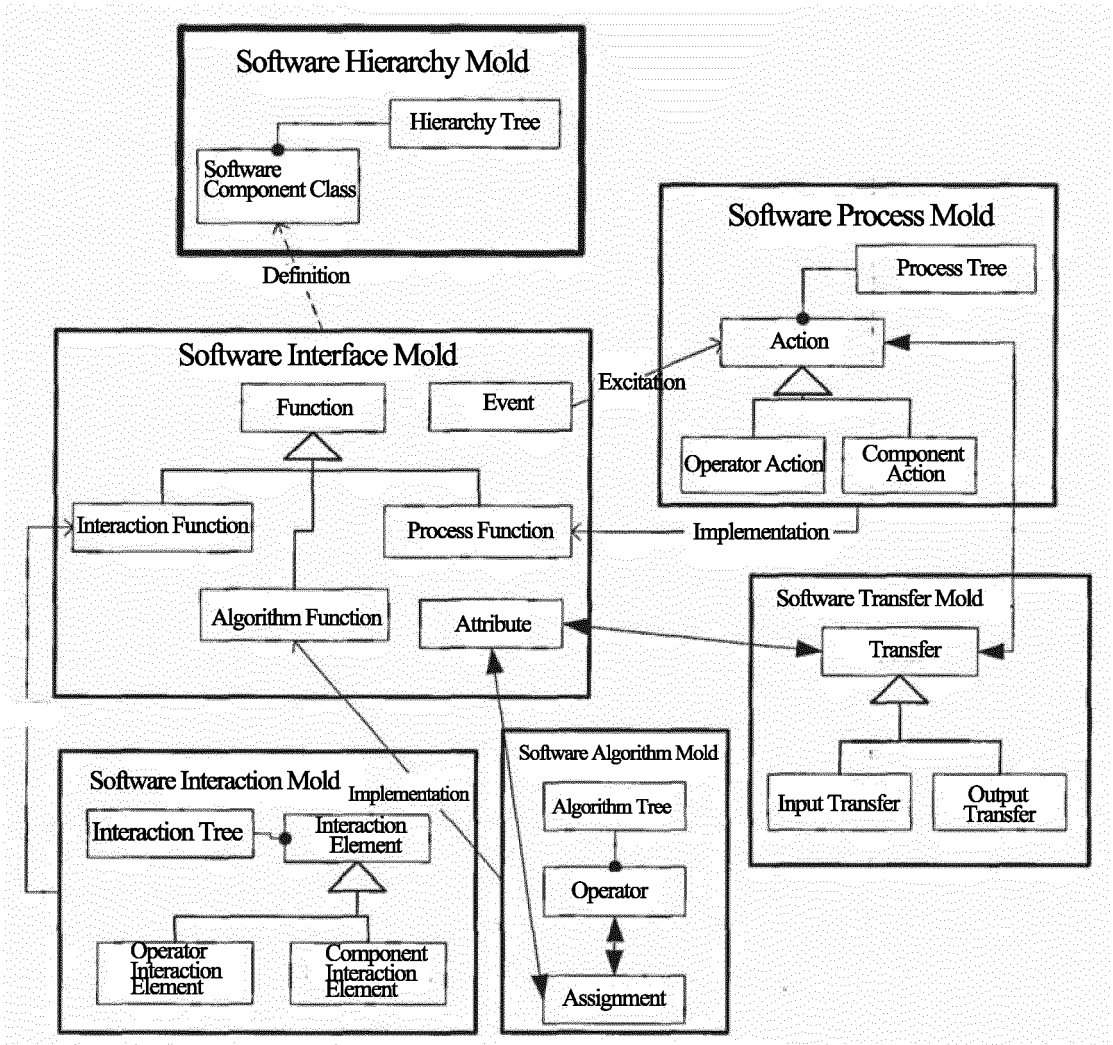

FIG. 1 Molds of the software meta model

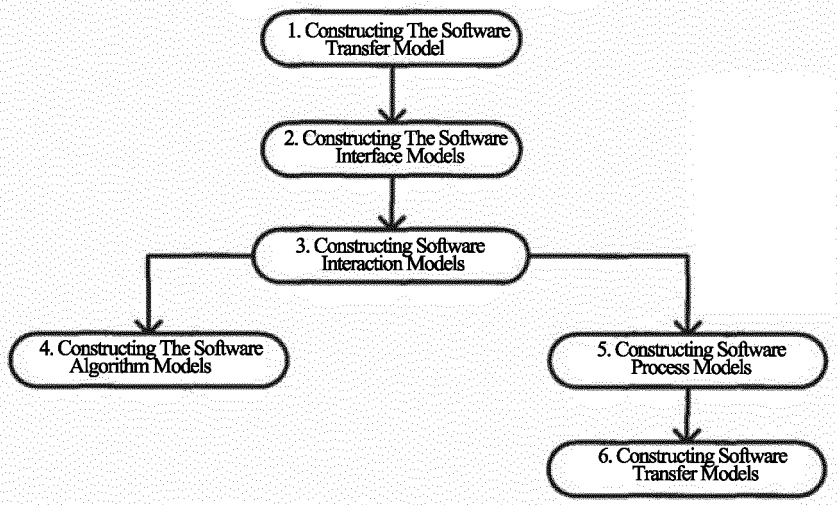

FIG. 2 Construction steps of a software model

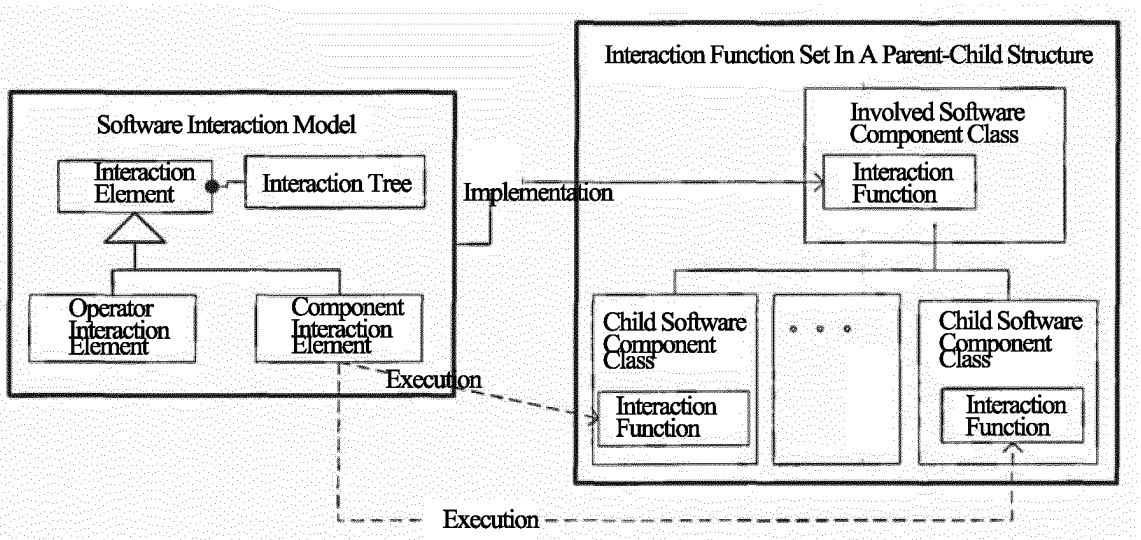

FIG. 3 Software interaction model

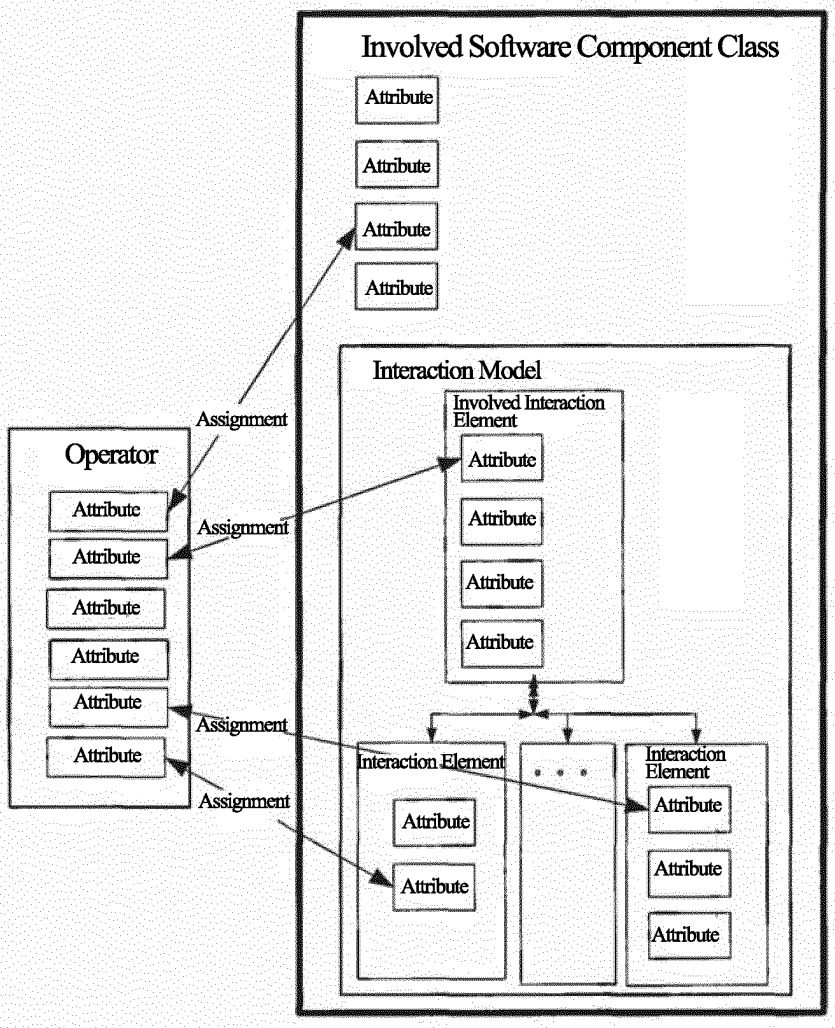

FIG. 4 Assignment rules in a software algorithm model

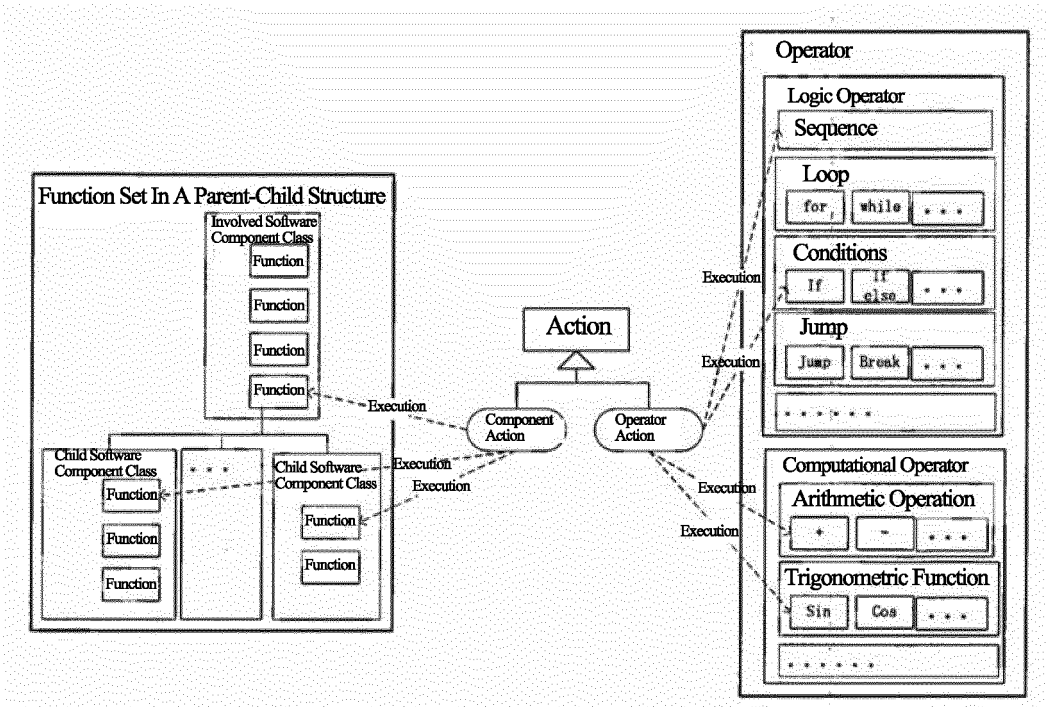

FIG. 5 Actions and function sets in a parent-child structure

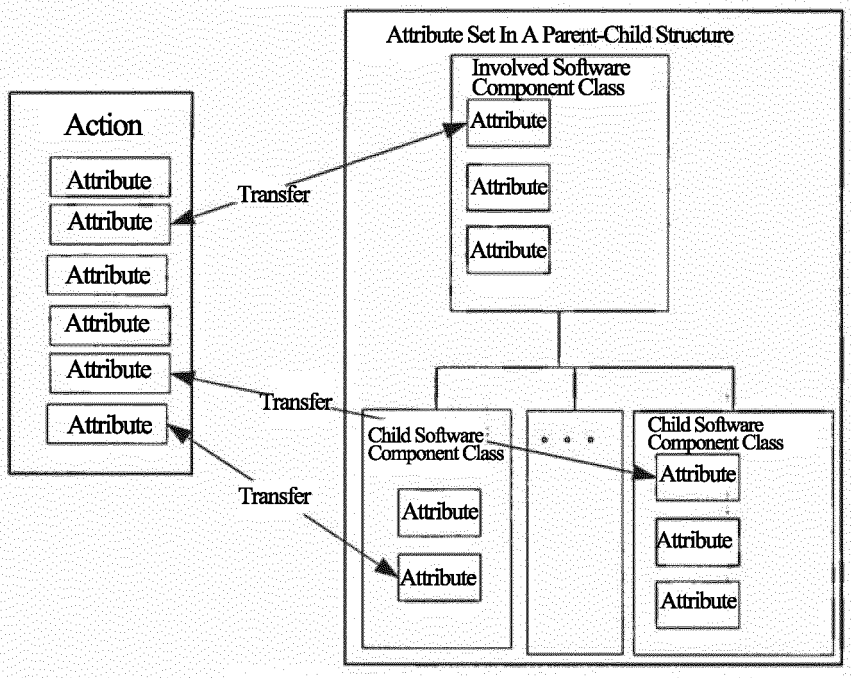

FIG. 6 Transfers and attribute sets in a parent-child structure

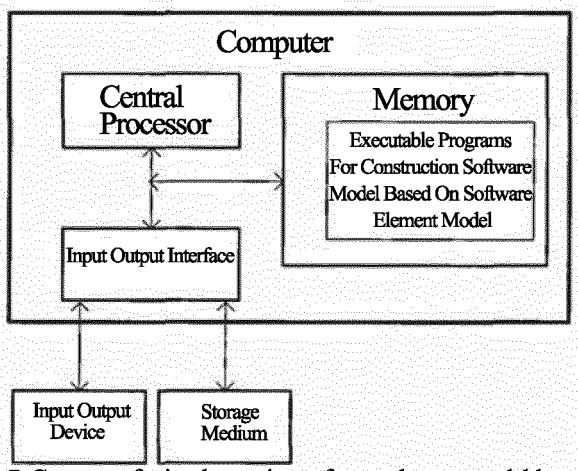

FIG. 7 Computer for implementing software element model-based universal modeling method for constructing software model

```
<Component Class  Name="Business Management YWGL"Instance Quantity="1">
  <Component Class  Name="Sales Management XSGL"  Instance Quantity="0">
    <Component Class  Name="Distributed Sales Product FXP"  Instance Quantity="1"/>
    <Component Class  Name="Direct Sales Product ZXP"  Instance Quantity="1"/>
  </Component Class>
  <Component Class  Name="Production Management SCGL"  Instance Quantity="0">
    <Component Class  Name="Main Parts ZJ"  Instance Quantity="1"/>
    <Component Class  Name="Auxiliary Parts LJ" Instance Quantity="1"/>
    <Component Class  Name="Finished Product CP"  Instance Quantity="1"/>
  </Component Class>
  <Component Class  Name="Purchase Management CGGL"  Instance Quantity="0"/>
  <Component Class  Name="Business Assistant YWZS"  Instance Quantity="1"/>
</Component Class>
```

FIG. 8 Business management YWGL hierarchy model

```
<Attribute Set>
    <Attribute  Name="Main Loop State"  Type="Bool"Value="True"/>
    <Attribute  Name="Business Operation State" Type="Bool"Value="True"/>
    <Attribute  Name="Production Product Type Quantity"  Type="Int"  Value="3"/>
    <Attribute  Name="Purchase Product Type Quantity"  Type="Int"  Value="2"/>
    <Attribute  Name="Sales Product Type Quantity"Type="Int"  Value="0"/>
</Attribute Set>
<Function Set>
    <Interaction Function Name="Business Display"/>
    <Process Function Name="Operation Display And Control"/>
    <Process Function Name="Main Business Procedure"/>
    <Process Function Name="Business Configuration"/>
    <Process Function Name="Business Operation"/>
    <Process Function Name="Instance Creation"/>
    <Process Function Name="Instance Configuration"/>
    <Process Function Name="Configuration Display And Control"/>
    <Algorithm Function Name="Product Type Configuration"/>
</Function Set>
```

FIG. 9 Business management YWGL interface model

```
< Attribute Set >
        < AttributeName = “Product Name”      Type = “String”   Value = “Sales Product” / >
        < AttributeName = “Product Serial Number”  Type = “Int”       Value = “1” / >
        < AttributeName = “Inventory Quantity”       Type = “Int”       Value = “0” / >
        < AttributeName = “Minimum Inventory Quantity” Type = “Int”       Value = “0” / >
        < AttributeName = “Contract Quantity” Type = “Int”       Value = “0” / >
        < AttributeName = “Receipt Quantity”  Type = “Int”       Value = “0” / >
        < AttributeName = “Order Quantity”    Type = “Int”       Value = “0” / >
        < AttributeName = “Shipment Quantity”       Type = “Int”       Value = “0” / >
        < AttributeName = “Total Shipment Quantity”       Type = “Int”       Value = “0” / >
        < AttributeName = “Demand Quantity” Type = “Int”       Value = “0” / >
< / Attribute Set >
< Function Set >
        < Interaction Function Name = “Sales Display” / >
        < Process Function     Name = “Sales Display And Control”     Mode = “Event” / >
        < Process Function     Name = “Sales Operation” / >
        < Algorithm Function Name = “Sales Update” / >
< / Function Set >
```

FIG. 10 Sales management XSGL interface model

```
< Attribute Set >
    < Attribute   Name = “Product Name” Type = “String” Value = “Self-Developed Product” / >
    < Attribute   Name = “Product Serial Number”  Type = “Int”       Value = “0” / >
    < Attribute   Name = “Order Quantity”    Type = “Int”       Value = “0” / >
    < Attribute   Name = “Processed Quantity”       Type = “Int”       Value = “0” / >
    < Attribute   Name = “Delivery Quantity” Type = “Int”       Value = “0” / >
    < Attribute   Name = “Total Delivery Quantity” Type = “Int”       Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function    Name = “Production Display” / >
    < Process Function        Name = “Production Display And Control”     Mode = “Event” / >
    < Process Function        Name = “Production Operation” / >
    < Process Function        Name = “Production Planning” / >
    < Process Function        Name = “Production Implementation” / >
    < Process Function        Name = “Production Delivery” / >
    < Algorithm Function      Name = “Production Update” / >
< / Function Set >
```

FIG. 11 Production management SCGL interface model

```
< Attribute Set >
    < Attribute   Name = “Product Name”      Type = “String”   Value = “Purchase Product” / >
    < Attribute   Name = “Product Serial Number”  Type = “Int”       Value = “0” / >
    < Attribute   Name = “Pending Purchase Quantity”    Type = “Int”       Value = “0” / >
    < Attribute   Name = “Purchased Quantity”       Type = “Int”       Value = “0” / >
    < Attribute   Name = “Delivery Quantity” Type = “Int”       Value = “0” / >
    < Attribute   Name = “Total Delivery Quantity” Type = “Int”       Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function    Name = “Purchase Display” / >
    < Process Function        Name = “Purchase Display And Control”       Mode “Event” / >
    < Process Function        Name = “Purchase Operation” / >
    < Algorithm Function      Name = “Purchase Implementation” / >
    < Algorithm Function      Name = “Purchase Update” / >
< / Function Set >
```

FIG. 12 Purchase management CGGL interface model

```
< Attribute Set >
    < Attribute   Name = “Product Serial Number”  Type = “Int”       Value = “0” / >
    < Attribute   Name = “Constant Zero”      Type = “Int”       Value = “0” / >
    < Attribute   Name = “Comparison Result”       Type = “Bool”     Value = “True” / >
< / Attribute Set >
```

FIG. 13 Business assistant YWZS interface model

```
< Attribute Set >
    < Attribute  Name = “Contract Quantity” Type = “Int”  Value = “0” / >
    < Attribute  Name = “Shipment Quantity”  Type = “Int”  Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function  Name = “Distributed Sales Display” / >
    < Process Function  Name = “Distributed Sales Display And Control” Mode = “Event” / >
    < Algorithm Function  Name = “Distributed Sales Update” / >
< / Function Set >
```

FIG. 14 Distributed sales product FXP interface model

```
< Attribute Set >
    < Attribute  Name = “Contract Quantity” Type = “Int”  Value = “3” / >
    < Attribute  Name = “Shipment Quantity”  Type = “Int”  Value = “4” / >
< / Attribute Set >
< Function Set >
    < Interaction Function  Name = “Direct Sales Display” / >
    < Process Function  Name = “Direct Sales Display And Control”  Mode = “Event” / >
    < Algorithm Function  Name = “Direct Sales Update” / >
< / Function Set >
```

FIG. 15 Direct sales product ZXP interface model

```
< Attribute Set >
    < Attribute  Name = “Pending Processing Quantity”  Type = “Int”  Value = “0” / >
    < Attribute  Name = “Processed Quantity”  Type = “Int”  Value = “0” / >
    < Attribute  Name = “Delivery Quantity” Type = “Int”  Value = “0” / >
    < Attribute  Name = “Total Delivery Quantity” Type = “Int”  Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function  Name = “Main Parts Display” / >
    < Algorithm Function  Name = “Main Parts Processing” / >
    < Algorithm Function  Name = “Main Parts Delivery” / >
    < Algorithm Function  Name = “Main Parts Update” / >
< / Function Set >
```

FIG. 16 Main parts ZJ interface model

```
< Attribute Set >
    < Attribute  Name = “Pending Processing Quantity”  Type = “Int”  Value = “0” / >
    < Attribute  Name = “Processed Quantity”  Type = “Int”  Value = “0” / >
    < Attribute  Name = “Delivery Quantity” Type = “Int”  Value = “0” / >
    < Attribute  Name = “Total Delivery Quantity” Type = “Int”  Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function  Name = “Auxiliary Parts Display” / >
    < Algorithm Function  Name = “Auxiliary Parts Processing” / >
    < Algorithm Function  Name = “Auxiliary Parts Delivery” / >
< Algorithm Function  Name = “Auxiliary Parts Update” / >
< / Function Set >
```

FIG. 17 Auxiliary parts LJ interface model

```
< Attribute Set >
    < Attribute  Name = “Pending Processing Quantity”  Type = “Int”      Value = “0” / >
    < Attribute  Name = “Processed Quantity”      Type = “Int”      Value = “0” / >
    < Attribute  Name = “Single Set Main Parts Quantity”      Type = “Int”      Value = “2” / >
    < Attribute  Name = “Single Set Auxiliary Parts Quantity” Type = “Int”      Value = “6” / >
    < Attribute  Name = “Main Parts Inventory”    Type = “Int”      Value = “0” / >
    < Attribute  Name = “Main Parts Receipt Quantity”  Type = “Int”      Value = “0” / >
    < Attribute  Name = “Auxiliary Parts Inventory”      Type = “Int”      Value = “0” / >
    < Attribute  Name = “Auxiliary Parts Receipt Quantity”    Type = “Int”      Value = “0” / >
< / Attribute Set >
< Function Set >
    < Interaction Function   Name = “Finished Product Display” / >
    < Algorithm Function    Name = “Part Receipt” / >
    < Algorithm Function    Name = “Finished Product Assembly” / >
    < Algorithm Function    Name = “Finished Product Update” / >
< / Function Set >
```

FIG. 18 Finished product CP interface model

```
< Window Body  ID = “Business Management Window Body” >
  < Free Layout  ID = “Business Management Root Layout”    Width = “Auto”  Height = “515” >
        < Tag  ID = “Business Management Interface”  Text Content = “Business Management
        Interface”    Width = “Auto”  Height = “Auto” / >
        < Stack Layout    ID = “Business Configuration Stack Layout”  Width = “Auto”
         Height = “Auto” >
              < Tag       Background Color = “Gray” Text Content = “Production Product
              Type Quantity ” Width = “Auto”  Height = “Auto” / >
              < Textbox ID = “Production Product Type Quantity Textbox”  Text Content =
              “3”  Width = “100”    Height = “20” / >
              < Tag       Background Color = “Gray” Text Content = “Purchase Product Type
              Quantity ” Width = “Auto”  Height = “30” / >
              < Textbox ID = “Purchase Product Type Quantity Textbox”      Text Content =
              “2”  Width = “100”    Height = “20” / >
              < Stack Layout  Width = “50”      Height = “Auto” / >
              < Button   ID = “Business Configuration Button”   Text Content = “Business
              Configuration”  Width = “70”      Height = “25” / >
        < / Stack Layout >
        < Card >
              < Card Page      ID = “Production Management Card Page”     Page Name =
              “Production Management” >
                    < Free Layout ID = “Production Management Card Page Root” >
                          < Component Interaction       ID = “Production Display “
                            Component Class = “Production Management SCGL”
                            Interaction Function = “Production Display” / >
                    < / Free Layout >
              < / Card Page >
              < Card Page      ID = “Purchase Management Card Page”       Page Name =
              “Purchase Management” >
                    < Free Layout ID = “Purchase Management Card Page Root” >
                          < Component Interaction       ID = “Purchase Display”
                            Component Class = “Purchase Management CGGL”
                            Interaction Function = “Purchase Display” / >
                    < / Free Layout >
              < / Card Page >
              < Card Page      ID = “Sales Management Card Page”     Page Name = “Sales
              Management” >
                    < Free Layout ID = “Sales Management Card Page Root” >
                          < Component Interaction       ID = “Sales Display”   Component
                          Class = “Sales Management XSGL”       Interaction Function =
                          “Sales Display” / >
                    < / Free Layout >
              < / Card Page >
        < / Card >
  < / Free Layout >
< / Window Body >
```

FIG. 19 Business display interaction model

```
< Class template ID = “Sales Instance Group Layout” Width = “Auto”Height = “Auto” Direction = “Longitudinal” >
    < Instance Template ID = “Sales Instance” Width = “Auto”Height = “Auto” >
        < Stack Layout ID = “Sales Instance” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
            < Stack Layout ID = “Sales Product Name” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
                < Tag ID = “Sales Product Name” Width = “Auto”Height = “Auto” Text Content = “Sales Product” / >
                < Tag ID = “Sales Product Serial Number” Width = “Auto”Height = “Auto” Text Content = “1” / >
            < / Stack Layout >
            < Stack Layout ID = “Sales Data” Width = “Auto”Height = “Auto” Direction = “Longitudinal” >
                < Stack Layout ID = “Sales Contract Quantity” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
                    < Tag ID = “Contract Quantity” Background Color = “Gray” Width = “Auto”Height = “Auto”
                    Text Content = “Contract Quantity” / >
                    < Tag ID = “Sales Contract Quantity Display” Width = “50” Height = “Auto”. Text Content”0” / >
                < / Stack Layout >
                < Stack Layout ID = “Sales Receipt Quantity” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “Auto” Text Content = “Receipt
                    Quantity” / >
                    < Tag ID = “Sales Receipt Quantity Display” Width = “50” Height = “Auto” Text Content = “0”
                    / >
                < / Stack Layout >
                < Stack Layout ID = “Sales Shipment Quantity” Width = “Auto”Height = “Auto” Direction =
                “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “30” Text Content = “Shipment Quantity”
                    / >
                    < Tag ID = “Sales Shipment Quantity Display” Width = “70” Height = “Auto” Text Content
                    = “0” / >
                < / Stack Layout >
                < Stack Layout ID = “Sales Total Shipment Quantity” Width = “Auto”Height = “Auto” Direction =
                “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “Auto” Text Content = “Total
                    Shipment Quantity” / >
                    < Tag ID = “Sales Total Shipment Quantity Display” Width = “70” Height = “Auto” Text Content
                    = “0” / >
                < / Stack Template” >
            < / Stack Layout >
            < Component Interaction ID = “Sales Display” Component Class = “Distributed Sales Product FXP” Interaction Function
            = “Sales Display” / >
            < Component Interaction ID = “Sales Display” Component Class = “Direct Sales Product ZXP” Interaction Function
            = “Sales Display” / >
        < / Stack Template >
    < / Instance Template >
< / Class template >
```

FIG. 20 Sales display interaction model

```
< Class template ID = “Production Instance Group Layout” Width = “Auto”Height = “Auto” Direction = “Longitudinal” >
    < Instance Template ID = “Production Instance” Width = “Auto”Height = “Auto” >
        < Stack Layout ID = “Production Instance” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
            < Stack Layout ID = “Production Product Name” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
                < Tag ID = “Production Product Name” Width = “Auto”Height = “Auto” Text Content = “Production
                Product” / >
                < Tag ID = “Production Product Serial Number” Width = “Auto”Height = “Auto” Text Content = “1”
                / >
            < / Stack Layout >
            < Stack Layout ID = “Production Data” Width = “Auto”Height = “Auto” Direction = “Longitudinal” >
                < Stack Layout ID = “Production Order Quantity” Width = “Auto”Height = “Auto” Direction =
                “Horizontal” >
                    < Tag ID = “Order Quantity” Background Color = “Gray” Width = “Auto”Height = “Auto” Text
                    Content = “Order Quantity” / >
                    < Tag ID = “Production Order Quantity Display” Width = “50” Height = “Auto” Text Content
                    = “0” / >
                < / Stack Layout >
                < Stack Layout ID = “Production Completion” Width = “Auto”Height = “Auto” Direction = “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “Auto” Text Content = “Processed
                    Quantity” / >
                    < Tag ID = “Production Completion Quantity Display” Width = “50 = “Height = “Auto” Text
                    Content = “0” / >
                < / Stack Layout >
                < Stack Layout ID = “Production Delivery Quantity” Width = “Auto”Height = “Auto” Direction =
                “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “30” Text Content = “Delivery Quantity”
                    / >
                    < Tag ID = “Production Delivery Quantity Display” Width = “70” Height = “Auto” Text Content
                    = “1” / >
                < / Stack Layout >
                < Stack Layout ID = “Production Total Delivery Quantity” Width = “Auto”Height = “Auto” Direction =
                “Horizontal” >
                    < Tag Background Color = “Gray” Width = “Auto”Height = “Auto” Text Content = “Total
                    Delivery Quantity” / >
                    < Tag ID = “Production Total Delivery Quantity Display” Width = “70” Height = “Auto” Text
                    Content = “0” / >
                < / Stack Template” >
            < / Stack Layout >
            < Button ID = “Production Completion” Content = “Production Completion” / >
            < Component Interaction ID = “Main Parts Display” Component Class = “Main Parts ZJ” Interaction Function = “Main
            Parts Display” / >
            < Component Interaction ID = “Auxiliary Parts Display” Component Class = “Auxiliary Parts LJ” Interaction Function
            = “Auxiliary Parts Display” / >
            < Component Interaction ID = “Finished Product Display” Component Class = “Finished Product CP” Interaction
            Function = “Finished Product Display” / >
        < / Stack Template” >
    < / Instance Template >
< / Class template >
```

FIG. 21 Production display interaction model

```
< Class template ID = “Purchase Instance Group Layout” Width = “Auto” Height = “Auto” Direction =
“Longitudinal” >
  < Instance Template >
    < Stack Layout ID = “Purchase Instance” Width = “Auto” Height = “Auto” Direction =
    “Horizontal” >
        < Stack Layout ID = “Purchase Product Name” Width = “Auto” Height = “Auto” Direction
        = “Horizontal” >
            < Tag ID = “Purchase Product Name” Width = “Auto” Height = “Auto” Text Content =
            “Purchase Product” / >
            < Tag ID = “Purchase Product Serial Number” Width = “32” Height = “30” Text
            Content = “0” / >
        < Stack Layout ID = “Purchase Data” Width = “Auto” Height = “Auto” Direction =
        “Longitudinal” >
            < Stack Layout ID = “pending purchasing Quantity” Width = “Auto” Height =
            “Auto” >
                < Tag ID = “Pending Purchase Quantity” Background Color = “Gray” Width =
                “Auto” Height = “30” Text Content = “Pending Purchase Quantity” / >
                < Tag ID = “pending purchasing Quantity Display” Width = “50” Height =
                “Auto” Text Content = “0” / >
            < / Stack Layout >
            < Stack Layout ID = “Purchased Quantity” Width = “Auto” Height = “Auto” >
                < Tag ID = “Purchased Quantity” Background Color = “Gray” Width = “Auto”
                Height = “Auto” Text Content = “Purchased Quantity” / >
                < Tag ID = “Purchased Quantity Display” Width = “50” Height = “30”
                Text Content = “0” / >
            < / Stack Layout >
            < Stack Layout ID = “Purchase Delivery Quantity” Width = “Auto” Height =
            “Auto” >
                < Tag ID = “Delivery Quantity” Background Color = “Gray” Width = “Auto”
                Height = “30” Text Content = “Delivery Quantity” / >
                < Tag ID = “Purchase Delivery Quantity Display” Width = “50” Height =
                “Auto” Text Content = “0” / >
            < / Stack Layout >
            < Stack Layout ID = “Purchase Total Delivery Quantity” Width = “Auto” Height =
            “Auto” >
                < Tag ID = “Total Delivery Quantity” Background Color = “Gray” Width =
                “Auto” Height = “Auto” Text Content = “Total Delivery Quantity” / >
                < Tag ID = “Purchase Total Delivery Quantity Display” Width = “100” Height =
                “30” Text Content = “0” / >
            < / Stack Layout >
        < / Stack Layout >
    < / Stack Layout >
    < Button ID = “Purchase Completion” Content = “Purchase Completion” / >
  < / Stack Layout >
< / Instance Template >
< / Class template >
```

FIG. 22 Purchase display interaction model

```
< Stack Layout ID = “Distributed Sales Product Root” Width = “Auto” Height = “Auto” >
    < Tag Width = “Auto” Height = “Auto” Text Content = “Information On Distributed Sales ” / >
    < Stack Layout ID = “Distributed Sales Contract Quantity” Width = “Auto” Height = “Auto”
      Direction = “Horizontal” >
      < Tag Background Color = “Gray” Width = “Auto” Height = “30” Text Content = “Contract
      Quantity” / >
      < Textbox ID = “Distributed Sales Contract Quantity” Width = “100” Height = “20”
        Text Content = “0” / >
    < / Stack Layout >
    < Stack Layout ID = “Distributed Sales Shipment Quantity” Width = “Auto” Height = “Auto”
      Direction = “Horizontal” >
      < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content = “Shipment
      Quantity” / >
      < Textbox ID = “Distributed Sales Shipment Quantity” Width = “100” Height = “20”
        Text Content = “0” / >
      < Button ID = “Distributed Sales Completion” Width = “Auto” Height = “25” Text
      Content = “Distributed Sales Completion” / >
    < / Stack Template >
< / Stack Layout >
```

FIG. 23 Distributed sales display interaction model

```
< Stack Layout ID = “Direct Sales Product Root” Width = “Auto” Height = “Auto” >
    < Tag Width = “Auto” Height = “30” Text Content = “Information On Direct Sales” / >
    < Stack Layout ID = “Direct Sales Contract Quantity” Width = “Auto” Height = “Auto” Direction =
    “Horizontal” >
        < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content = “Contract
        Quantity” / >
        < Textbox ID = “Direct Sales Contract Quantity Textbox” Width = “100” Height = “Auto” Text
        Content = “0” / >
    < / Stack Layout >
    < Stack Layout ID = “Direct Sales Shipment Quantity” Width = “Auto” Height = “Auto” Direction =
    “Horizontal” >
        < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content = “Shipment
        Quantity” / >
        < Textbox ID = “Direct Sales Shipment Quantity Textbox” Width = “100” Height = “Auto” Text
        Content = “0” / >
        < Button ID = “Direct Sales Completion” Width = “70” Height = “Auto” Text Content = “Direct Sales
        Completion” / >
    < / Stack Layout >
< / Stack Layout >
```

FIG. 24 Direct sales display interaction model

```
< Stack Layout ID = “Main Parts Root” Width = “Auto” Height = “Auto” >
    < Tag Width = “154” Height = “30” Text Content = “Information On Main Parts” / >
    < Stack Layout ID = “Main Parts Pending Processing Quantity” Width = “Auto” Height =
    “Auto” Direction = “Horizontal” >
        < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content = “Pending
        Processing Quantity” / >
        < Tag ID = “Main Parts Pending Processing Quantity” Width = “100” Height = “30”
        Text Content = “0” / >
        < Stack Layout ID = “Main Parts Processed Quantity” Width = “Auto” Height = “Auto”
        Direction = “Horizontal” >
            < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content =
            “Processed Quantity” / >
            < Tag ID = “Main Parts Processed Quantity” Width = “Auto” Height = “Auto”
            Text Content = “0” / >
        < / Stack Layout >
        < Stack Layout ID = “Main Parts Delivery Quantity” Width = “Auto” Height = “Auto”
        Direction = “Horizontal” >
            < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content =
            “Delivery Quantity” / >
            < Tag ID = “Main Parts Delivery Quantity” Width = “Auto” Height = “Auto”
            Text Content = “0” / >
        < / Stack Layout >
    < / Stack Layout >
< / Stack Layout >
```

FIG. 25 Main parts display interaction model

```
< Stack Layout ID = “Auxiliary Parts Root” Width = “Auto” Height = “Auto” >
    < Tag Width = “154” Height = “30” Text Content = “Information On Auxiliary Parts” / >
    < Stack Layout ID = “Auxiliary Parts Pending Processing Quantity”Width = “Auto” Height =
    “Auto”Direction = “Horizontal” >
        < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content = “Pending
        Processing Quantity” / >
        < Tag ID = Auxiliary Parts Pending Processing Quantity Display” Width = “100” Height =
        “30” Text Content = “0” / >
        < Stack Layout ID = “Auxiliary Parts Processed Quantity” Width = “Auto” Height =
        “Auto” Direction = “Horizontal” >
            < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content =
            “Processed Quantity” / >
            < Tag ID = “Auxiliary Parts Processed Quantity Display” Width = “100” Height = “30”
            Text Content = “0” / >
        < / Stack Layout >
        < Stack Layout ID = “Auxiliary Parts Delivery Quantity” Width = “Auto” Height =
        “Auto” Direction = “Horizontal” >
            < Tag Background Color = “Gray” Width = “Auto” Height = “Auto” Text Content =
            “Delivery Quantity” / >
            < Tag ID = “Auxiliary Parts Delivery Quantity Displays” Width = “100” Height = “30”
            Text Content = “0” / >
        < / Stack Layout >
    < / Stack Layout >
< / Stack Layout >
```

FIG. 26 Auxiliary parts display interaction model

```
< Stack Layout   ID = “Finished Product Root”      Width = “Auto”  Height = “Auto” >
      < Tag      Width = “154”    Height = “30”    Text Content = “Information On Finished Product” / >
      < Stack Layout  ID = “Finished Product Pending Processing Quantity Stack”     Width = “Auto”
           Height = “Auto” Direction = “Horizontal” >
           < Stack Layout  ID = “Stack Layout”   Width = “Auto”  Height = “Auto” >
                < Tag      Background Color = “Gray” Width = “Auto”  Height = “Auto” Text Content =
                “Pending Processing Quantity” / >
                < Tag      ID = “Finished Product Pending Processing Quantity Display”  Width = “100”
                     Height = “30”    Text Content = “0” / >
           < / Stack Layout >
           < Stack Layout  ID = “Finished Product Processed Quantity”   Width = “Auto”  Height =
           “Auto”     Direction = “Horizontal” >
                < Tag      Background Color = “Gray” Width = “Auto”  Height = “Auto” Text Content =
                “Processed Quantity” / >
                < Tag      ID = “Finished Product Processed Quantity Display”      Width = “100”
                     Height = “30”    Text Content = “0” / >
           < / Stack Layout >
      < / Stack Layout >
< / Stack Layout >
```

FIG. 27 Finished product display interaction model.

```
< Algorithm >
      < Assignment Operator      ID = “Production Product Type Configuration”
           Input = “Production Product Type Quantity Textbox. Text Content”   Output = “Production
           Product Type Quantity ”
/ >
      < Assignment Operator      ID = “Purchase Product Type Configuration”
           Input = “Purchase Product Type Quantity Textbox. Text Content”     Output = “Purchase
           Product Type Quantity ”
      / >
      < Addition Operator   ID = “Sales Product Type Configuration”
           Augend = “Production Product Type Quantity Textbox. Text Content”
           Addend = “Purchase Product Type Quantity Textbox. Text Content”
           Summation = “Sales Product Type Quantity”
       / >
< / Algorithm >
```

FIG. 28 Product type configuration algorithm model

```
< Algorithm >
      < Assignment Operator      ID = “Sales Contract Quantity Update”
           Input = “Contract Quantity” Output = “Sales Contract Quantity Display Tag. Text Content”
      / >
      < Assignment Operator      ID = “Sales Receipt Quantity Update”
           Input = “Receipt Quantity”  Output = “Sales Receipt Quantity Display Tag. Text Content”
      / >
      < Assignment Operator      ID = “Sales Shipment Quantity Update”
           Input = “Shipment Quantity”      Output = “Sales Shipment Quantity Display Tag. Text
           Content”
      / >
      < Assignment Operator      ID = “Sales Total Shipment Quantity Update”
           Input = “Total Shipment Quantity”      Output = “Sales Total Shipment Quantity Display Tag.
           Text Content”
      / >
< / Algorithm >
```

FIG. 29 Sales update algorithm model

```
< Algorithm >
    < Assignment Operator  ID = “Production Completion Delivery Assignment”
         Input = “Processed Quantity”  Output = “Delivery Quantity”
    / >
    < Addition Operator      ID = “Production Total Delivery Quantity”
         Augend = “Assignment1.Output”    Addend = “Total Delivery Quantity”      Summation = “Total
         Delivery Quantity”
    / >
    < Subtraction Operator  ID = “Production Completion Reset”
         Minuend = “Assignment1.Output”  Subtrahend = “Processed Quantity”      Margin = “Processed
         Quantity”
    / >
< / Algorithm >
```

FIG. 30 Production delivery algorithm model

```
< Algorithm >
    < Assignment Operator  ID = “Production Order Quantity Update”
        Input = “Order Quantity”      Output = “Production Order Quantity Display Tag. Text Content”
    / >
    < Assignment Operator  ID = “Production Completion Quantity Update”
        Input = “Processed Quantity”  Output = “Production Completion Quantity Display Tag. Text
        Content”
    / >
    < Assignment Operator  ID = “Production Delivery Quantity Update”
        Input = “Delivery Quantity”   Output = “Production Delivery Quantity Display Tag. Text Content”
    / >
    < Assignment Operator  ID = “Production Total Delivery Quantity Update”
        Input = “Total Delivery Quantity”   Output = “Production Total Delivery Quantity Display Tag.
        Text Content”
    / >
< / Algorithm >
```

FIG. 31 Production update algorithm model

```
< Algorithm >
    < Assignment Operator  ID = “Purchased Quantity Assignment”
        Input = “Pending Purchase Quantity”      Output = “Purchased Quantity”
    / >
    < Assignment Operator  ID = “Purchase Delivery Quantity Assignment”
        Input = “Pending Purchase Quantity”      Output = “Delivery Quantity”
    / >
    < Addition Operator      ID = “Purchase Total Delivery Quantity Summary”
        Augend = “Pending Purchase Quantity”  Addend = “Total Delivery Quantity”      Summation =
        “Total Delivery Quantity”
    / >
    < Subtraction Operator  ID = “pending purchasing Quantity Reset”
        Minuend = “Pending Purchase Quantity”  Subtrahend = “Pending Purchase Quantity”      Margin
        = “Pending Purchase Quantity”
    / >
< / Algorithm >
```

FIG. 32 Purchase implementation algorithm model

```
< Algorithm >
    < Assignment Operator      ID = “pending purchasing Quantity Update”
        Input = “Pending Purchase Quantity”   Output = “pending purchasing Quantity Display Tag.
        Text Content”
    / >
    < Assignment Operator      ID = “Purchased Quantity Update”
        Input = “Purchased Quantity”      Output = “Purchased Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Purchase Delivery Quantity Update”
        Input = “Delivery Quantity” Output = “Purchase Delivery Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Purchase Total Delivery Quantity Update”
        Input = “Total Delivery Quantity” Output = “Purchase Total Delivery Quantity Display Tag. Text
        Content”
    / >
< / Algorithm >
```

FIG. 33 Purchase update algorithm model

```
< Algorithm >
    < Assignment Operator  ID = “Distributed Sales Contract Quantity Recording”
            Input = “Distributed Sales Display. Distributed Sales Contract Quantity Textbox. Text Content”
                Output = “Contract Quantity”
    / >
    < Assignment Operator  ID = “Distributed Sales Shipment Quantity Recording”
            Input = “Distributed Sales Display. Distributed Sales Shipment Quantity Textbox. Text Content”
                Output = “Shipment Quantity”
    / >
< / Algorithm >
```

FIG. 34 Distributed sales update algorithm model

```
< Algorithm >
    < Assignment Operator      ID = “Direct Sales Contract Quantity Recording”
        Input = “Direct Sales Display. Direct Sales Contract Quantity Textbox. Text Content”  Output =
        “Contract Quantity”
    / >
    < Assignment Operator      ID = “Direct Sales Shipment Quantity Recording”
        Input = “Direct Sales Display. Direct Sales Shipment Quantity Textbox. Text Content” Output =
        “Shipment Quantity”
    / >
< / Algorithm >
```

FIG. 35 Direct sales update algorithm model

```
< Algorithm >
    < Assignment Operator      ID = “Main Parts Pending Processing Quantity Update”
        Input = “Main Parts Pending Processing Quantity”  Output = “Main Parts Pending Processing
        Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Main Parts Processed Quantity Update”
        Input = “Processed Quantity”      Output = “Main Parts Display. Main Parts Processed Quantity
        Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Main Parts Delivery Quantity Update”
        Input = “Delivery Quantity”      Output = “Main Parts Display. Main Parts Delivery Quantity
        Display Tag. Text Content”
    / >
< / Algorithm >
```

FIG. 36 Main parts update algorithm model

```
< Algorithm >
    < Assignment Operator      ID = “Auxiliary Parts Pending Processing Quantity Update”
        Input = “Pending Processing Quantity”  Output = “Auxiliary Parts Display. Auxiliary Parts
        Pending Processing Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Auxiliary Parts Processed Quantity Update”
        Input = “Processed Quantity”      Output = “Auxiliary Parts Display. Auxiliary Parts Processed
        Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Auxiliary Parts Delivery Quantity Update”
        Input = “Delivery Quantity” Output = “Auxiliary Parts Display. Auxiliary Parts Delivery
        Quantity Display Tag. Text Content”
    / >
< / Algorithm >
```

FIG. 37 Auxiliary parts update algorithm model

```
< Algorithm >
    < Assignment Operator      ID = “Finished Products Pending Processing Quantity Update”
        Input = “Finished Products Pending Processing Quantity”      Output = “Finished Product
        Display. Finished Products Pending Processing Quantity Display Tag. Text Content”
    / >
    < Assignment Operator      ID = “Finished Products Processed Quantity Update”
        Input = “Processed Quantity”      Output = “Finished Product Display. Finished Products
        Processed Quantity Display Tag. Text Content”
    / >
< / Algorithm >
```

FIG. 38 Finished product update algorithm model

```
< Algorithm >
    < Assignment Operator    ID = “Main Parts Pending Processing / Processed Assignment”
        Input = “Pending Processing Quantity”  Output = “Processed Quantity”
    / >
    < Subtraction Operator    ID = “Main Parts Pending Processing Quantity Reset”
        Minuend = “Pending Processing Quantity”   Subtrahend = “Pending Processing Quantity”
            Margin = “Pending Processing Quantity”
    / >
< / Algorithm >
```

FIG. 39 Main parts processing algorithm model

```
< Algorithm >
    < Assignment Operator    ID = “Main Parts Processed Delivery”
        Input = “Processed Quantity”    Output = “Delivery Quantity”
    / >
    < Addition Operator  ID = “Main Parts Total Delivery Quantity”
        Augend = “Processed Quantity”  Addend = “Total Delivery Quantity”    Summation = “Total
        Delivery Quantity”
    / >
    < Subtraction Operator    ID = “Main Parts Processed Reset”
        Minuend = “Processed Quantity”  Subtrahend = “Processed Quantity”    Margin =
        “Processed Quantity”
    / >
< / Algorithm >
```

FIG. 40 Main parts delivery algorithm model

```
< Algorithm >
    < Assignment Operator    ID = “Auxiliary Parts Pending Processing / Processed Assignment”
        Input = “Pending Processing Quantity”  Output = “Processed Quantity”
    / >
    < Subtraction Operator    ID = “Auxiliary Parts Pending Processing Reset”
        Minuend = “Pending Processing Quantity”   Subtrahend = “Pending Processing Quantity”
        Margin = “Pending Processing Quantity”
    / >
< / Algorithm >
```

FIG. 41 Auxiliary parts processing algorithm model

```
< Algorithm >
    < Assignment Operator    ID = “Auxiliary Parts Processed Delivery”
        Input = “Processed Quantity”    Output = “Delivery Quantity”
    / >
    < Addition Operator  ID = “Auxiliary Parts Total Delivery Quantity”
        Augend = “Processed Quantity”  Addend = “Total Delivery Quantity”    Summation = “Total
        Delivery Quantity”
    / >
    < Subtraction Operator    ID = “Auxiliary Parts Processed Reset”
        Minuend = “Processed Quantity”  Subtrahend = “Processed Quantity”    Margin =
        “Processed Quantity”
    / >
< / Algorithm >
```

FIG. 42 Auxiliary parts delivery algorithm model

```
< Algorithm >
    < Addition Operator  ID = “Main Parts Receipt”
        Augend = “Main Parts Inventory” Addend = “Main Parts Receipt Quantity”    Summation =
        “Main Parts Inventory”
    / >
    < Addition Operator  ID = “Auxiliary Parts Receipt”
        Augend = “Auxiliary Parts Inventory”  Addend = “Auxiliary Parts Receipt Quantity”
            Summation = “Auxiliary Parts Inventory”
    / >
< / Algorithm >
```

FIG. 43 Auxiliary parts receipt algorithm model

```
< Algorithm >
    < Multiplication Operator    ID = “Assembly Main Parts”
        Multiplicand = “Main Parts Pending Processing Quantity”    Multiplier = “Single Set Main
        Parts Quantity”
    / >
    < Subtraction Operator    ID = “Assembly Main Parts Inventory”
        Minuend = “Finished Product CP. Main Parts Inventory” Subtrahend = “Assembly Main Parts.
        Product”    Margin = “Finished Product CP. Main Parts Inventory”
    / >
    < Multiplication Operator    ID = “Assembly Auxiliary Parts”
        Multiplicand = “Auxiliary Parts Pending Processing Quantity “Multiplier = “Single Set
        Auxiliary Parts Quantity”
    / >
    < Subtraction Operator    ID = “Assembly Auxiliary Parts Inventory”
        Minuend = “Auxiliary Parts Receipt Quantity”    Subtrahend = “Assembly Auxiliary Parts.
        Product”    Margin = “Auxiliary Parts Inventory”
    / >
    < Assignment Operator    ID = “Finished Product Completion”
        Input = “Pending Processing Quantity” Output “Processed Quantity”
    / >
    < Subtraction Operator    ID = “Finished Products Pending Processing Reset”
        Minuend = “Pending Processing Quantity”    Subtrahend = “Pending Processing Quantity”
        Margin = “Pending Processing Quantity”
    / >
< / Algorithm >
```

FIG. 44 Finished product assembly algorithm model

```
< Process >
    < Component Action  ID = “Business Display”    Component ID = “Business Management YWGL”
        Component Function = “Business Display” / >
    < Frame Loop Action  ID = “Main Business Procedure Frame Loop” >
        < Condition Action    ID = “Main Procedure Condition” >
            < “True” Branch >
                < Component Action  ID = “Business Operation”    Component ID =
                “Business Management YWGL”
                    Component Function = “Business Operation” / >
                < Component Action  ID = “Operation Display And Control”  Component ID =
                “Business Management YWGL” / >
                    Component Function = “Operation Display And Control” / >
            < / “True” Branch >
            < “False” Branch >
                < Component Action  ID = “Configuration Display And Control”    Component ID
                = “Business Management YWGL” / >
                    Component Function = “Configuration Display And Control” / >
            < “False” Branch/ >
        < / Condition Action >
    < / Frame Loop Action >
< / Process >
```

FIG. 45 Main business procedure process model

```
< Process  Mode = “Event” >
    < Event Association    Event Source = “Business Configuration Button”    Event = “Mouse Click”
    ID = “Business Configuration”
        Component ID = “Business Management YWGL”  Component Function = “Business
        Configuration” / >
< / Process >
```

FIG. 46 Configuration display and control process model

```
< Process >
    < Component Action    ID = “Business Type Configuration”    Component ID = “Business
    Management YWGL”
            Component Function = “Type Configuration” / >
    < Component Action    ID = “Business Instance Creation” Component ID = “Business Management
    YWGL”
            Component Function = “Instance Creation” / >
    < Component Action    ID = “Business Instance Configuration” Component ID = “Business
    Management YWGL”
            Component Function = “Instance Configuration” / >
    < Negating Action ID = “Business Operation State Negating” / >
< / Process >
```

FIG. 47 Business configuration process model

```
< Process >
    < Instance Creation Action    ID = “Sales Instance Creation” / >
    < Instance Creation Action    ID = “Production Instance Creation” / >
    < Instance Creation Action    ID = “Purchase Instance Creation” / >
< / Process >
```

FIG. 48 Business instance creation process model

```
< Process >
    < Traversal Action ID = “Production Configuration Traversal” >
        < Increment Action      ID = “Production Serial Number Increment” / >
        < Assignment Action     ID = “Production Serial Number Assignment” / >
    < / Traversal Action >
    < Traversal Action ID = “Purchase Configuration Traversal” >
        < Increment Action      ID = “Purchase Serial Number Increment” / >
        < Assignment Action     ID = “Purchase Serial Number Assignment” / >
    < / Traversal Action >
    < Assignment Action     ID = “Sales Serial Number Reset” / >
    < Traversal Action ID = “Sales Configuration Traversal” >
        < Increment Action      ID = “Sales Serial Number Increment” / >
        < Assignment Action     ID = “Sales Serial Number Assignment” / >
        < Traversal Action ID = “Sales-Production Configuration Traversal” >
            < Consistency Comparison Action   ID = “Sales-Production Configuration Comparison” / >
            < Condition Action      ID = “Sales-Production Configuration Condition” >
                < “True” Branch >
                    < Assignment Action     ID = “Sales Production Product Name Assignment” / >
                < / “True” Branch >
                < “False” Branch/ >
            < / Condition Action >
        < / Traversal Action >
        < Traversal Action ID = “Sales-Purchase Configuration Traversal” >
            < Consistency Comparison Action   ID = “Sales-Purchase Configuration Comparison” / >
            < Condition Action      ID = “Sales-Purchase Configuration Condition” >
                < “True” Branch >
                    < Assignment Action     ID = “Sales-Purchase Product Name Assignment” / >
                < / “True” Branch >
                < “False” Branch/ >
            < / Condition Action >
        < / Traversal Action >
    < / Traversal Action >
< / Process >
```

FIG. 49 Business instance configuration process model

```
< Process >
    < Traversal Action ID = “Sales Operation Traversal” >
        < Traversal Action ID = “Sales-Production Operation Traversal” >
            < Consistency Comparison Action  ID = “Sales-Production Operation Comparison” / >
            < Condition Action  ID = “Sales-Production Operation Condition” >
                < “True” Branch >
                    < Component Action  ID = “Sales-Production Operation”
                        Component ID = “Sales Management XSGL”  Component Function =
                        “Sales Operation” / >
                < / “True” Branch >
                < “False” Branch/ >
            < / Condition Action >
        < / Traversal Action >
        < Traversal Action ID = “Sales-Purchase Operation Traversal” >
            < Consistency Comparison Action  ID = “Sales-Purchase Operation Comparison” / >
            < Condition Action  ID = “Sales-Purchase Operation Condition” >
                < “True” Branch >
                    < Component Action  ID = “Sales-Purchase Operation”
                        Component ID = “Sales Management XSGL”  Component Function =
                        “Sales Operation” / >
                < / “True” Branch >
                < “False” Branch/ >
            < / Condition Action >
        < / Traversal Action >
    < / Traversal Action >
< / Process >
```

FIG. 50 Business operation process model

```
< Process >
    < Traversal Action  ID = “Sales Display And Control Traversal” >
        < Component Action  ID = “Sales Display And Control” Component ID = “Sales Management
        XSGL”  Component Function = “Sales Display And Control” / >
    < / Traversal Action >
    < Traversal Action ID = “Production Display And Control Traversal” >
        < Component Action  ID = “Production Display And Control” Component ID = “Production
        Management SCGL”  Component Function = “Production Display And Control” / >
    < / Traversal Action >
    < Traversal Action ID = “Purchase Display And Control Traversal” >
        < Component Action  ID = “Purchase Display And Control”  Component ID = “Purchase
        Management CGGL”  Component Function = “Purchase Display And Control” / >
    < / Traversal Action >
< / Process >
```

FIG. 51 Operation display and control process model

```
< Process >
    < Component Action  ID = “Distributed Sales Display And Control” Component ID = “Distributed
    Sales Product FXP”  Component Function = “Distributed Sales Display And Control” / >
    < Component Action  ID = “Direct Sales Display And Control”  Component ID = “Direct Sales
    Product ZXP”  Component Function = “Direct Sales Display And Control” / >
    < Component Action  ID = “Sales Update”  Component ID = “Sales Management XSGL”
        Component Function = “Sales Update” / >
< / Process >
```

FIG. 52 Sales display and control process model

```
< Process >
    < Addition Action  ID = “Sales Receipt” / >
    < Addition Action  ID = “Shipment Quantity Summary” / >
    < Addition Action  ID = “Total Shipment Quantity Summary” / >
    < Addition Action  ID = “Inventory Quantity Summary” / >
    < Addition Action  ID = “Contract Quantity Summary” / >
    < Addition Action  ID = “Demand Quantity Summary” / >
    < Addition Action  ID = “Order Quantity Summary” / >
< / Process >
```

FIG. 53 Sales operation process model

```
< Process  Mode = “Event” >
      < Event Association   Event Source = “Production Completion Button”     Event = “Mouse Click”
                  ID = “Production Operation”Component ID = “Production Management SCGL”
                  Component Function = “Production Operation” / >
< / Process >
```

FIG. 54 Production display and control process model

```
< Process >
      < Component Action  ID = “Production Planning”  Component ID = “Production Management
      SCGL”    Component Function = “Production Planning” / >
      < Component Action  ID = “Production Implementation”Component ID = “Production Management
      SCGL”    Component Function = “Production Implementation” / >
      < Component Action  ID = “Production Delivery”  Component ID = “Production Management
      SCGL”    Component Function = “Production Delivery” / >
      < Component Action  ID = “Production Update”    Component ID = “Production Management
      SCGL”    Component Function = “Production Update” / >
      < Component Action  ID = “Main Parts Update”    Component ID = “Main Parts ZJ”  Component
      Function = “Main Parts Update” / >
      < Component Action  ID = “Auxiliary Parts Update”     Component ID = “Auxiliary Parts LJ”
            Component Function = “Auxiliary Parts Update” / >
      < Component Action  ID = “Finished Product Update”    Component ID = “Finished Product CP”
            Component Function = “Finished Product Update” / >
< / Process >
```

FIG. 55 Production operation process model

```
< Process >
      < Multiplication Action      ID = “Main Parts Pending Processing Quantity Summary” / >
      < Multiplication Action      ID = “Auxiliary Parts Pending Processing Quantity Summary” / >
< / Process >
```

FIG. 56 Production planning process model

```
< Process >
      < Component Action  ID = “Main Parts Processing”      Component ID = “Main Parts ZJ”
            Component Function = “Main Parts Processing” / >
      < Component Action  ID = “Main Parts Delivery”  Component ID = “Main Parts ZJ”  Component
      Function = “Production Delivery” / >
      < Component Action  ID = “Auxiliary Parts Processing” Component ID = “Auxiliary Parts LJ”
            Component Function = “Auxiliary Parts Processing” / >
      < Component Action  ID = “Auxiliary Parts Delivery “Component   ID = “Auxiliary Parts LJ”
            Component Function = “Auxiliary Parts Delivery” / >
      < Component Action  ID = “Auxiliary Parts Receipt”    Component ID = “Finished Product CP”
            Component Function = “Auxiliary Parts Receipt” / >
      < Component Action  ID = “Finished Product Assembly”       Component ID = “Finished Product
      CP”  Component Function = “Finished Product Assembly” / >
< / Process >
```

FIG. 57 Production implementation process model

```
< Process  Mode = “Event” >
      < Event Association   Event Source = “Purchase Completion Button”      Event = “Mouse Click”
                  ID = “Purchase Operation”  Component ID = “Purchase Management CGGL”
                  Component Function = “Purchase Operation” / >
< / Process >
```

FIG. 58 Purchase display and control process mode

```
< Process >
      < Component Action  ID = “Purchase Implementation”  Component ID = “Purchase Management
      CGGL”    Component Function = “Purchase Implementation” / >
      < Component Action  ID = “Purchase Update”     Component ID = “Purchase Management CGGL”
            Component Function = “Purchase Update” / >
< / Process >
```

FIG. 59 Purchase operation process model

```
< Process Mode = "Event" >
      < Event Association      Event Source = "Distributed Sales Completion Button "Event
      = "Mouse Click"
            ID = "Distributed Sales Update"   Component ID = "Distributed Sales Product
            FXP" Component Function = "Distributed Sales Update" / >
< / Process >
```

FIG. 60 Distributed sales display and control process model

```
< Process Mode = "Event" >
      < Event Association      Event Source = "Direct Sales Completion Button"      Event =
      "Mouse Click"
            ID = "Direct Sales Update"   Component ID = "Direct Sales Product ZXP"
                  Component Function = "Direct Sales Update" / >
< / Process >
```

FIG. 61 Direct sales display and control process model

```
< Frame Loop Action     ID = "Main Procedure Frame Loop"
      < Input Transfer   State = "Business Management YWGL. Main Loop State" / >
/ >
```

FIG. 62 Main procedure frame loop transfer model

```
< Condition Action ID = "Main Procedure Condition"
      < Input Transfer   State = "Business Management YWGL. Business Operation
      State" / >
/ >
```

FIG. 63 Main procedure condition transfer model

```
< Negating Action  ID = "Business Operation State Negating"
      < Input Transfer   Input = "Business Management YWGL. Business Operation
      State" / >
      < Output Transfer      Output = "Business Management YWGL. Business
      Operation State" / >
/ >
```

FIG. 64 Business operation state negation transfer model

```
< Instance Creation Action     ID = "Sales Instance Creation"
    < Input Transfer     Type = "Sales Management XSGL" / >
    < Input Transfer     Quantity = "Business Management YWGL. Sales Product Type
    Quantity" / >
/ >
```

FIG. 65 Sales instance creation transfer model

```
< Instance Creation Action     ID = "Production Instance Creation"
    < Input Transfer     Type = "Production Management SCGL" / >
    < Input Transfer     Quantity = "Business Management YWGL. Production Product
    Type Quantity " / >
/ >
```

FIG. 66 Production instance creation transfer model

```
< Instance Creation Action     ID = "Purchase Instance Creation"
      < Input Transfer   Type = "Purchase Management CGGL" / >
      < Input Transfer   Quantity = "Business Management YWGL. Purchase Product
      Type Quantity " / >
/ >
```

FIG. 67 Purchase instance creation transfer model

```
< Traversal Action  ID = "Production Configuration Traversal"
      < Input Transfer   Type = "Production Management SCGL" / >
/ >
```

FIG. 68 Production configuration traversal transfer model

```
< Increment Action ID = "Production Serial Number Increment"
      < Input Transfer   Input = "Business Assistant YWZS. Product Serial Number" / >
      < Output Transfer       Output = "Business Assistant YWZS. Product Serial
      Number" / >
/ >
```

FIG. 69 Production serial number increment transfer model

```
< Assignment Action      ID = "Production Serial Number Assignment"
      < Output Transfer       Output = "Production Management SCGL. Product Serial
      Number" / >
      < Input Transfer   Input = "Business Assistant YWZS. Product Serial Number" / >
/ >
```

FIG. 70 Production serial number assignment transfer model

```
< Traversal Action  ID = "Purchase Configuration Traversal"
      < Input Transfer   Type = "Purchase Management CGGL" / >
/ >
```

FIG. 71 Purchase configuration traversal transfer model

```
< Assignment Action      ID = "Purchase Serial Number Assignment"
      < Output Transfer       Output = "Purchase Management CGGL. Product Serial
      Number" / >
      < Input Transfer   Input = "Business Assistant YWZS. Product Serial Number" / >
/ >
```

FIG. 72 Purchase serial number assignment transfer model

```
< Assignment Action      ID = "Sales Serial Number Reset"
      < Output Transfer       Output = "Business Assistant YWZS. Product Serial
      Number" / >
      < Input Transfer   Input = "Business Assistant YWZS. Constant Zero" / >
/ >
```

FIG. 73 Sales serial number reset transfer model

```
< Traversal Action  ID = "Sales Configuration Traversal"
      < Input Transfer   Type = "Sales Management XSGL" / >
/ >
```

FIG. 74 Sales configuration traversal transfer model

```
< Assignment Action      ID = "Sales Serial Number Assignment"
      < Output Transfer       Output = "Sales Management XSGL. Product Serial
      Number" / >
      < Input Transfer   Input = "Business Assistant YWZS. Product Serial Number" / >
/ >
```

FIG. 75 Sales serial number assignment transfer model

```
< Consistency Comparison Action ID = "Sales-Production Configuration Comparison"
    < Input Transfer     Comparand = "Sales Management XSGL. Product Serial
    Number" / >
    < Input Transfer      Comparison = "Production Management SCGL. Product
    Serial Number" >
    < Output Transfer   Result = "Business Assistant YWZS. Comparison Result" / >
/ >
```

FIG. 76 Sales-production configuration comparison transfer model

```
< Condition Action ID = "Sales-Production Configuration Condition"
      < Input Transfer   State = "Business Assistant YWZS. Comparison Result" / >
/ >
```

FIG. 77 Sales-production configuration condition transfer model

```
< Assignment Action      ID = "Sales Production Product Name Assignment"
      < Output Transfer       Output = "Sales Management XSGL. Product Name" / >
      < Input Transfer   Input = "Production Management SCGL. Product Name" / >
/ >
```

FIG. 78 Sales-production product name assignment transfer model

```
< Consistency Comparison Action ID = "Sales-Purchase Configuration Comparison"
    < Input Transfer     Comparand = "Sales Management XSGL. Product Serial
    Number" / >
    < Input Transfer     Comparison = "Purchase Management CGGL. Product Serial
    Number" / >
    < Output Transfer   Result = "Business Assistant YWZS. Comparison Result" / >
/ >
```

FIG. 79 Sales-purchase configuration comparison transfer model

```
< Condition Action ID = "Sales-Purchase Configuration Condition"
      < Input Transfer     State = "Business Assistant YWZS. Comparison Result" / >
/ >
```

FIG. 80 Sales-purchase configuration condition transfer model

```
< Assignment Action      ID = "Sales-Purchase Product Name Assignment"
      < Input Transfer   Input = "Purchase Management CGGL. Product Name" / >
      < Output Transfer  Output = "Sales Management XSGL. Product Name" / >
/ >
```

FIG. 81 Sales-purchase product name assignment transfer model

```
< Component Action       ID = "Sales-Production Operation"
      < Input Transfer   Receipt Quantity = "Production Management SCGL. Delivery
      Quantity" / >
      < Output Transfer  Order Quantity = "Production Management SCGL. Order
      Quantity" / >
/ >
```

FIG. 82 Sales-production operation transfer model.

```
< Component Action       ID = "Sales-Purchase Operation"
      < Input Transfer   Receipt Quantity = "Purchase Management CGGL. Delivery
      Quantity" / >
      < Output Transfer  Order Quantity = "Purchase Management CGGL. Pending
      Purchase Quantity" / >
/ >
```

FIG. 83 Sales-purchase operation transfer model

```
< Addition Action   ID = "Sales Receipt"
      < Input Transfer   Augend = "Sales Management XSGL. Inventory Quantity" / >
      < Input Transfer   Addend = "Sales Management XSGL. Receipt Quantity" / >
      < Output Transfer  Summation = "Sales Management XSGL. Inventory
      Quantity" / >
/ >
```

FIG. 84 Sales receipt transfer model

```
< Addition Action   ID = "Shipment Quantity Summary"
      < Input Transfer   Augend = "Distributed Sales Product FXP. Shipment Quantity"
      / >
      < Input Transfer   Addend = "Direct Sales Product ZXP. Shipment Quantity" / >
      < Output Transfer  Summation = "Sales Management XSGL. Shipment
      Quantity" / >
/ >
```

FIG. 85 Shipment quantity summary transfer model

```
< Addition Action   ID = "Total Shipment Quantity Summary"
      < Input Transfer   Augend = "Sales Management XSGL. Shipment Quantity" / >
      < Input Transfer   Addend = "Sales Management XSGL. Total Shipment
      Quantity" / >
      < Output Transfer  Summation = "Sales Management XSGL. Total Shipment
      Quantity" / >
/ >
```

FIG. 86 Total shipment quantity summary transfer model

```
< Subtraction Action     ID = "Inventory Quantity Summary"
    < Input Transfer     Minuend = "Sales Management XSGL. Inventory Quantity" / >
    < Input Transfer     Subtrahend = "Sales Management XSGL. Shipment Quantity"
    / >
    < Output Transfer    Margin = "Sales Management XSGL. Inventory Quantity" / >
/ >
```

FIG. 87 Inventory quantity summary transfer model

```
< Addition Action      ID = "Contract Quantity Summary"
      < Input Transfer     Augend = "Distributed Sales Product FXP. Shipment Quantity" / >
      < Input Transfer     Addend = "Direct Sales Product ZXP. Shipment Quantity" / >
      < Output Transfer    Summation = "Sales Management XSGL. Total Shipment Quantity" / >
/ >
```

FIG. 88 Contract quantity summary transfer model

```
< Addition Action      ID = "Demand Quantity Summary"
      < Input Transfer     Augend = "Sales Management XSGL. Minimum Inventory Quantity" / >
      < Input Transfer     Addend = "Sales Management XSGL. Contract Quantity" / >
      < Output Transfer    Summation = "Sales Management XSGL. Demand Quantity" / >
/ >
```

FIG. 89 Demand quantity summary transfer model

```
< Subtraction Action   ID = "Order Quantity Summary"
      < Input Transfer     Minuend = "Sales Management XSGL. Demand Quantity" / >
      < Input Transfer     Subtrahend = "Sales Management XSGL. Inventory Quantity" / >
      < Output Transfer    Margin = "Sales Management XSGL. Order Quantity" / >
/ >
```

FIG. 90 Order quantity summary transfer model

```
< Multiplication Action      ID = "Main Parts Pending Processing Quantity Summary"
      < Input Transfer     Multiplicand = "Production Management SCGL. Order Quantity" / >
      < Input Transfer     Multiplier = "Finished Product CP. Single Set Main Parts Quantity" / >
      < Output Transfer    Product = "Main Parts ZJ. Main Parts Pending Processing Quantity" / >
/ >
```

FIG. 91 Main-parts pending processing quantity summary transfer model

```
< Multiplication Action      ID = "Auxiliary Parts Pending Processing Quantity Summary"
      < Input Transfer     Multiplicand = "Production Management SCGL. Order Quantity" / >
      < Input Transfer     Multiplier = "Finished Product CP. Single Set Auxiliary Parts Quantity"
      / >
      < Output Transfer    Product = "Auxiliary Parts LJ. Auxiliary Parts Pending Processing
      Quantity" / >
/ >
```

FIG. 92 Auxiliary parts pending processing quantity summary transfer model

```
< Component Action   ID = "Auxiliary Parts Receipt"
      < Input Transfer     Main Parts Receipt Quantity = "Main Parts ZJ. Delivery Quantity" / >
      < Input Transfer     Auxiliary Parts Receipt Quantity = "Auxiliary Parts LJ. Delivery
      Quantity" / >
/ >
```

FIG. 93 Parts receipt transfer model

```
< Component Action   ID = "Finished Product Assembly"
      < Output Transfer    Processed Quantity = "Production Management SCGL. Processed
      Quantity"/ >
/ >
```

FIG. 94 Finished product assembly transfer model

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2015/000454

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/44 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNPAT, CNKI, EPODOC, WPI: template, element model, unified modelling language, algorithm, hierarchy, process, interaction, UML, model, meta, software, layer, interface, transmit

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 101477462 A (SHANDONG INSPUR SOFTWARE CO., LTD.), 08 July 2009 (08.07.2009), description, page 2, line 11 to page 4, line 4	1-9
A	CN 103092594 A (KINGDEE SOFTWARE (CHINA) CO., LTD.), 08 May 2013 (08.05.2013), the whole document	1-9
A	CN 1794170 A (JILIN UNIVERSITY), 28 June 2006 (28.06.2006), the whole document	1-9
A	US 2008263085 A1 (MICROSOFT CORPORATION), 23 October 2008 (23.10.2008), the whole document	1-9
A	US 2009132211 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION), 21 May 2009 (21.05.2009), the whole document	1-9

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 August 2015 (15.08.2015)	**18 September 2015 (18.09.2015)**
Name and mailing address of the ISA/CN: State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No.: (86-10) 62019451	Authorized officer **KANG, Kai** Telephone No.: (86-10) **62414043**

Form PCT/ISA/210 (second sheet) (July 2009)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2015/000454

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 101477462 A	08 July 2009	None	
CN 103092594 A	08 May 2013	None	
CN 1794170 A	28 June 2006	CN 100580622 C	13 January 2010
US 2008263085 A1	23 October 2008	JP 2010525452 A	22 July 2010
		CN 101663663 A	03 March 2010
		CN 101663663 B	13 February 2013
		EP 2149094 A1	03 February 2010
		US 7765241 B2	27 July 2010
		WO 2008130768 A1	30 October 2008
US 2009132211 A1	21 May 2009	TW 200935249 A	16 August 2009
		CN 101441563 A	27 May 2009

Form PCT/ISA/210 (patent family annex) (July 2009)

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- CN 200610125050 **[0005]**